TECHNISCHE UNIVERSITÄT WIEN

VIENNA UNIVERSITY OF TECHNOLOGY

Dissertation

Architecture and Design of Service Platforms for Next Generation Telecommunication Networks

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Doktors der technischen Wissenschaften unter der Leitung von

Univ. Prof. Dr. Harmen R. van As
Institut für Breitbandkommunikation
Technische Universität Wien
Favoritenstraße 9/388, A-1040 Wien, Österreich

und

Univ. Prof. Dr. Schahram Dustdar
Information Systems Institute
Technische Universität Wien
Argentinierstrasse 8/184-1, A-1040 Wien, Österreich

eingereicht an der Technischen Universität Wien
Fakultät für Elektrotechnik und Informationstechnik
von

Dipl.-Ing. Rudolf Pailer
Matr.Nr. E8632288

Wien, im Oktober 2007

I dedicate this work to my beloved family,
Ulrike, Maximilian and Marie.

panta rhei – everything flows
Plato, 428 – 348 BC

Table of Contents

Acknowledgements

This thesis was done under the supervision of Univ. Prof. Dr. Harmen R. van As, Institute of Broadband Communications, Vienna University of Technology. I thank Univ. Prof. Dr. Harmen R. van As for his professional contribution to the research and for his generous support throughout my studies.
I would also like to thank Univ. Prof. Dr. Schahram Dustdar, Information Systems Institute, Vienna University of Technology, who agreed to evaluate this thesis.
I further want to take the opportunity to thank all those people directly and indirectly involved in this thesis and to show my appreciation of all the assistance I have received throughout my work.
In particular I would like to thank Dr. Sandford Bessler, Dr. Peter Reichl, Dipl.-Ing. Joachim Fabini, and Mag. Dipl.-Ing. Rainer Huber for their encouragement, support and numerous valuable discussions.
Part of the work presented in this thesis is based on research carried out in the ftw. projects “Service Platforms”, “Service Platform and Interoperability”, “LoL@ UMTS Applications Development”, “SIMS Services in IMS”, and “CAIPIRINA Convergence toward All-IP: IMS Realization Issues for NGN Applications” (see also www.ftw.at). ftw. is co-financed by the Austrian government’s Kplus Competence Centers Program, the City of Vienna, and the participating companies.

Contributions

There are four main contributions that are described in this thesis. First a service platform architecture for next generation networks is developed that blends the characteristics and requirements of classical telecommunication networks with the concepts and design principles of service oriented computing. A prototype of the proposed architecture has been implemented and is described in detail. Second the validity of the architectural approach is proven, by showing how value added services in general and location based services in particular can be mapped to the proposed design patterns. Third a patent of the author is presented, that shows how the proposed architecture can be used in combination with cryptographical methods to ensure the privacy of service platform users. Fourth the performance of the proposed location service architecture is analysed by simulation and by the methods of queueing theory.
This thesis is structured in five chapters, where the first two chapters *1 Introduction* and *2 Service Platforms* give a general overview on service platforms in next generation networks and the following three chapters *3 Service Platform Architecture and Design*, *4 Location Based Services*, *5 Performance Analysis of a Location Presence System* contain the afore listed contributions.
The chapters containing the contributions are structured in sections, where sections presenting the current state-of-the-art, related work and standards are followed by section where contributions of the author are presented. In order to give the reader some orientation, where the main contributions of this thesis can be found, the following list of chapters and sections shows all sections with contributions of the author in *italic print*:

The results of the research groups that the author has been part of have been published on several occasions. The following list presents the relations of published work to sections in this thesis:

- The introduction of a service platform for the implementation and operation of multimedia services in next generation networks is proposed and the new technologies are presented and evaluated with the goal to specify a carrier-grade service framework for multimedia communications in heterogeneous networks [Bessler].
  (see Section 2.6)
- A service platform architecture complying with Service Oriented Architecture (SOA) principles is presented. A mapping of Session Initiation Protocol (SIP) functionality to Parlay services is proposed [Pailer 01], [Pailer 03] and a prototype implementation using the SIP Servlet API is described.
  (see Sections 3.3, 3.5)
- The separation of service logic from network technology is proposed by using the Parlay APIs to build a communication switch as a "softswitch" application running on a service platform [Stadler].
  (see Section 3.6)
- An efficient scheme has been developed and patented (Austrian patent number A 363/2004) that enables localisation of users in next generation networks by third party application while ensuring their privacy [Pailer Pat 04] [Jorns].
  (see Section 4.4)
- A novel location architecture for the 3GPP IP Multimedia Subsystem is proposed, using SIP based application layer signaling which supports both terminal generated and network-calculated location information [Pailer 05], [Ratti], [Pailer 06], [Fabini], [Reichl 06a], [Reichl 06b], [Gartner].
  (see Section 4.5)
- It is the main advantage of the proposed terminal-based location enabler that it supports triggered location updates in a simple and scalable way, so that necessary location updates are kept to an absolute minimum and scarce resources in the radio access network are used in the most economic way. The gains in network capacity are estimated in a detailed performance analysis by using the methods of traffic simulation and queueing theory [Reichl 06b].
  (see Sections 5.2, 5.3, 5.4)

1 Introduction

Applications in next generation telecommunication networks offer services to customers that are based on the available network resources and capabilities. The term *next generation* refers to telecommunication networks that are heterogeneous regarding the network technology as well as the offered services.

Next generation networks offer a broad range of multimedia services, such as voice, video, messaging, web, streaming, broadcast, gaming, virtual realities, workflow management. These services are offered for and have to be adapted to very different network access technologies like GSM, GPRS, UMTS, WLAN, Bluetooth, or Ethernet. Furthermore the user will access those services on terminals that vary in parameters like size, computing resources, battery capacity, or operating systems.

The ITU-T Study Group 13 [ITU-T NGN] defines the term “Next Generation Network” as follows:

> A Next Generation Network (NGN) is a packet-based network able to provide services including Telecommunication Services and able to make use of multiple broadband, QoS-enabled transport technologies and in which service-related functions are independent from underlying transport-related technologies. It offers unrestricted access by users to different service providers. It supports generalized mobility which will allow consistent and ubiquitous provision of services to users.

A NGN is characterized by several fundamental aspects. First a NGN is based packet-based transport network. Control functions among bearer capabilities, call/session, and application/ service are cleanly separated. Open interfaces are used and these interfaces are abstracted of network technology. A wide range of services and applications is supported and implementations are based on service building blocks. These services include real time/ streaming/ non-real time services and multi-media services that require different end-to-end QoS. The user shall be able to move between physical locations and access networks while remaining seamlessly attached to a service. Finally a NGN shall provide interworking with legacy networks via open interfaces.

Service platforms for next generation telecommunication networks offer a framework for the creation, execution and management of services and applications in heterogeneous telecommunication networks. Given the heterogeneity of network access technology, service requirements and user terminals, a service platform offers an abstract view on the available terminal capabilities, network resources and back-end systems. Thus end-user services and applications can be built around platform services and platform components. It is extremely important that service and business logic are implemented on an abstract service platform model that hides the underlying network technologies. Otherwise the complexity of application development will reach a level that is not manageable technically and organizationally.

The inherent heterogeneity of next generation networks has also the consequence that services and applications have to be developed for rather small user groups and that demand for services changes rapidly. This segmentation and fluctuation of the service offering requires a fast, flexible and cost effective service development and service operation process. Service platforms meet these demands by supporting the whole service lifecycle. Service platforms give application developers a set of re-usable service components that can be easily assembled, integrated and tested. Service operators are furthermore able to configure, deploy and monitor applications based on a common service platform management layer.

The implementation of new mobile communication technologies will allow accessing the Internet not only from a PC but also via mobile phones, smartphones and other devices. New applications will emerge, combining several basic services like voice telephony, e-mail, voice over IP, mobility or web-browsing, and thus wiping out the borders between the fixed telephone network, mobile radio and the Internet. Offering those value-added services will be the key factor for success of network and service providers in an increasingly competitive market. On their way towards implementing the Next Generation Network (NGN), many mobile and fixed network operators have already started to migrate their telecommunication networks towards an All-IP infrastructure where voice loses its predominance and becomes just one among many services. The IP Multimedia Subsystem (IMS) [Camarillo2006], standardized by the 3rd Generation Partnership Project [3GPP], is the most promising candidate for replacing legacy, voice-dedicated mobile networks with an All-IP technology. As opposed to traditional IP-based networks, the IMS guarantees end-to-end Quality of Service (QoS) in the network and creates an infrastructure that enables the fast deployment of new IP-based services and flexible billing while still maintaining compatibility with existing applications. Additionally, the new horizontal service architecture of IMS bridges the traditional divide between circuit-switched and packet-switched networks, consolidating both sides into one single network for all services, and thus will have an immense impact on the whole way future mobile business applications are designed, developed and deployed.
Besides the mentioned flexibility of the IMS charging architecture, another vital advantage of IMS relates to easy integration of novel services and applications, simplifying the whole development cycle and shortening the time-to-market considerably. Here, service enablers exposing network functionality to external service providers are the cornerstones of modern service architectures, not only with respect to IMS, but also for the Parlay group or the Open Mobile Alliance (OMA).
This thesis is structured in three parts. The first part consists of the first three chapters *1 Introduction*, *2 Service Platforms*, *3 Service Platform Architecture and Design* and covers the general aspects of service platforms for next generation telecommunications networks. After an introduction, service platform architecture is discussed in detail and then the design of a concrete prototype is presented.
The second part of this thesis, Chapter *4 Location Based Services*, chooses Location Based Services (LBS) as a showcase in order to discuss architecture and design of value added services in service platforms. Location Based Services (LBS) are services that require information about the physical position of a user in order to provide 'added-value' to services in a Third Generation (3G) network. Location data may be plain geographical coordinates, access point cell ids, civil location in form of postal addresses or more abstract definitions like 'in the office', 'at home'. Example services are a map showing the user's current location or triggering a switch of the user profile when entering a specified area. This thesis proposes a novel privacy mechanism that is targeted for location and presence services in the 3G service architecture and uses cryptographic techniques well suited to run in small devices with little computing and power resources. Furthermore a terminal-based location enabler based on the IP Multimedia Subsystem presence architecture is proposed.
The third part finally, Chapter *5 Performance Analysis of a Location Presence System*, presents a detailed performance analysis for a carrier-grade deployment of a Location Service (LCS) based on the proposed presence location architecture. The methods of traffic simulation in combination with queuing theory are applied to models of a location presence system in order to investigate the system performance parameters.

2 Service Platforms

Service platforms offer a distributed framework for the creation, execution and management of services and applications in heterogeneous telecommunication networks. A service platform offers an abstracted view on the network resources and back-end systems that services and applications are based on.

The last decade has seen the tremendous success of two key technologies in telecommunications, the Internet or IP based data networks and cellular telephone networks. It is only a logical consequence to blend the flexibility of packet oriented data networks and the mobility of circuit switched radio networks into a third generation network that offers the best of both worlds. The third generation partnership project (3GPP) has been founded to define this new technology that will enable a wealth of new converged services.

In order to implement those services the 3GPP has defined a framework according to the Open Service Architecture [OSA] that includes the Parlay APIs [Parlay] for the control of multimedia services. 3GPP has also adopted the Session Initiation Protocol [SIP] protocol stack for the signaling of session control in an IP based core network.

During the projects “A2-Service Platforms and Interoperability”, our research group at the Telecommunications Research Center Vienna [FTW] proposed the introduction of a service platform for the implementation and operation of multimedia services in heterogeneous networks [Bessler] and evaluated the proposed new technologies with the goal to specify a carrier-grade service framework for multimedia communications in heterogeneous networks.

This chapter examines the business-roles of service platform users and identifies typical use cases and actors were. Furthermore several common building blocks for developing services for the Internet are described and a Service Oriented Architecture (SOA) approach for a service component-based, distributed platform for multimedia services is given.

2.1 Service Platform Actors

The actors in the business model of a Service Platform (SP) were first suggested and discussed by the Telecommunications Information Networking Architecture Consortium (TINA-C), that published specifications on service architecture, network resource architecture, distributed processing environments and reference points with the goal to define a common architecture for information and telecommunication services. For a more detailed overview see [Mampaey 00a], [TINA-C]. The Parlay group [Parlay] has built heavily on the TINA-C architecture when specifying their Parlay API and Parlay X Web Services. For a more detailed discussion on Parlay, see below.

Based on the proposed architectures by TINA-C and Parlay, the following business roles were identified:

- End user
- Subscriber
- Service provider
- Third party service provider
- Network provider

Figure 2-1 shows the identified Service Platform actors and use cases.

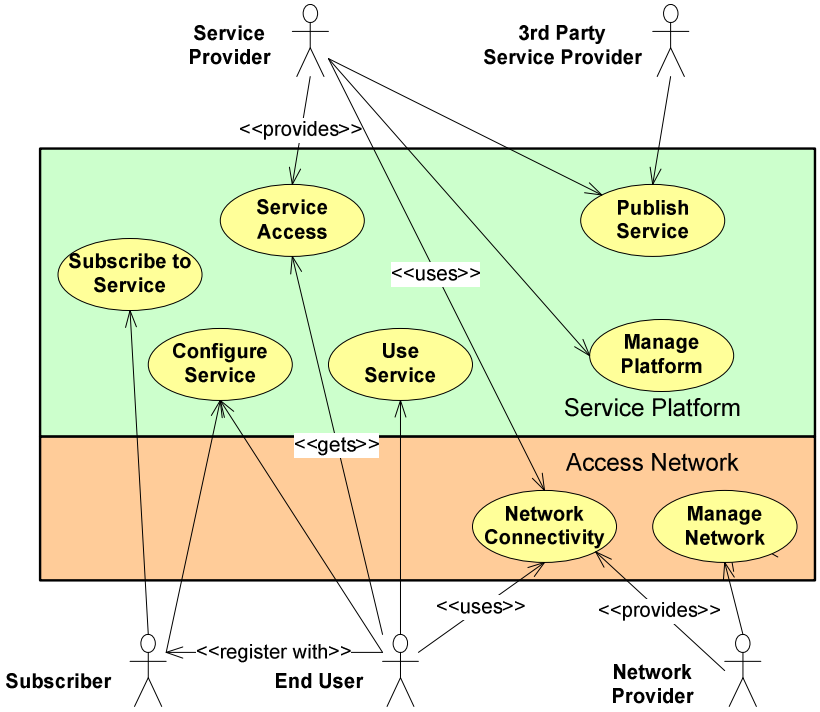


Figure 2-1: Service Platform actors and use cases

The service provider runs and maintains the service platform. He maintains the data store with all the users and service logic as well as the contracts and configuration of the subscribers. Furthermore he creates new services, which are deployed on the platform and offered to the subscribers. Also he provides connectivity and service level agreement for network access that is managed by the network provider.

The SP has to provide mechanisms for registration of subscriber resources such as content and web-servers, as well as enterprise resources needed for the implementation of a certain service.

A subscriber is a user or a company that pays for a service offered by the service provider. Subscribers do neither host services nor have to maintain the service platform.

End users are registered on the behalf of the subscriber. The representatives of the subscriber (operators) have a subscription side management application to configure and manage the service of their users.
Depending on the service, a registered user can configure an application by himself, such as call forwarding, or the task can be done via subscriber side management, e.g., in call center scenarios.
A third party service provider provides services and content, which is offered by the Service Platform. A service provider could be third party service provider too.

2.2 Service Platform Architecture

The idea of a service platform for telecommunication networks goes back to typical Intelligent Network (IN) systems, the most popular service architecture for the PSTN. The IN architecture provides a Service Creation Environment (SCE) in which a service is created from a set of standardized service-independent building blocks (SIBs).
[Gbaguidi] proposes a service platform for so called hybrid services, which is based on a Service Creation Environment (SCE) and a Service Infrastructure (SI). In the SCE service logic is put together by the service creator using a set of service components as building blocks. The service logic is then sent to a service factory, which creates a new instance of a service. The service instance is organized into service subsystems that are further up-loaded into the system infrastructure, more specifically in the controllers of the system elements.
The Service Infrastructure is composed of a collection of controllers and the underlying system elements of the Intelligent Network, such as terminals, gateways, network nodes. An event interface enables the data flow between the system elements and the service logic.
[Manione 99a] describes a service platform based on a TINA architecture that has been simplified to cope with the requirements of internet-telecom service development: simplified session modelling, identity of the service provider business role with the retailer, enhanced subscriber management, logical separation between service logic (service environment) and generic platform services.
The WebCentric web call centre application [Manione 99b] provides added value by offering VoIP call together with pushed WebPages to the caller, or other media combinations such as chat or email. The paper describes clearly the technology used for the different terminal types, without addressing generic interfaces to IN, or to go into the aspects of service creation and deployment.
The Open Mobile Appliance (OMA) has been founded to “facilitate global user adoption of mobile data services by specifying market driven mobile service enablers that ensure service interoperability across devices, geographies, service providers, operators, and networks, while allowing businesses to compete through innovation and differentiation”. OMA’s work has been largely concentrated on defining guidelines and specifications for enablers and services like WAP or Push-to-Talk. Recently OMA has also intensified their work on service architecture and service environments, by setting in place the OMA Architecture Working Group, who is responsible for the definition of the overall OMA system architecture [OMA]. The OMA architecture working group concentrates on

- Policy Evaluation Enforcement Management (PEEM)
  The PEEM enabler evaluates and/or enforces policies. Policies are applied to requests/response interactions with network. PEEM architecture gives a framework for defining, managing, evaluating, and enforcing policies in a way that is scalable and flexible yet independent of any specific implementation scheme. [OMA PEEM]
- Life Cycle Management (Deployment, Upgrade, Monitoring), Service Level Tracing and Service Model & Catalogue [OMA OSPE]

OMA has defined the following architecture principles [OMA AP]:

- OMA Principle Number 1 – Produce application enabler specifications that are independent of underlying networks, devices, operating systems and programming languages.
  OMA specifications must be network technology agnostic and must be applicable

across a wide range of underlying network technologies. While OMA specifications should not limit applications to use the lowest common denominator of available network features, the use of optimisations related to specific underlying technologies should not be required.

- OMA Principle Number 2 - Leverage existing standards.
- OMA Principle Number 3 – Provide open, interoperable, scalable, extensible, modular enabler specifications.
- OMA Principle Number 4 – Provide for service adaptability based on device capabilities, network characteristics and user preference.

The OMA has identified the OSA/Parlay APIs as a concrete implementation of their architectural requirements.

The service platform architecture proposed in this thesis is based on the principles of a Service Oriented Architecture (SOA). "Separation of concerns" is one of the core principles of Service-Oriented Architectures and is as well an established method in software engineering, where large and complex systems are broken down into a set of individual concerns that lead to smaller and less complex subsystems that are related to each other. A SOA approach implements individual concerns as autonomous service components that are loosely coupled to applications. The following separation principles are a main contribution of the TINA-C architecture [Mampaey 00b] [TINA-C]:

- **First TINA Separation Principle – "The service architecture shall be independent of the network architecture"**
  In classical PSTN networks, development of new telecommunication services was hindered by the fact that new services had to be integrated with the network protocols. With every new service the complexity of service integration grew exponentially. By introducing interfaces to an abstract network resource and connectivity system, services are separated from the underlying network technology. Thus the complexity of service integration is reduced considerably and service logic becomes independent of network technology.
- **Session Concept – "Separate session control from media control"**
  Multi-media session may involve a set of media channels (voice, video, whiteboard-control, game-control, instant-messaging) with very different requirements regarding QoS parameters like channel capacity, packet delay, delay jitter or error rate. A session is a managing context that is modeled independently of the media channels that are controlled by this session.
- **Second TINA Separation Principle – "Separate service access from service usage"**
  The session concept is used to differ between a service access session and a service session. The access session enables a clean separation of Authentication, Authorization and Accounting (AAA) functionality from the actual service usage. An access session context can furthermore be used by services to implement different service behavior based on the access type (e.g., reacting on bandwidth requirements or roaming scenarios). Finally an access session enables the concept of presence.
  The service session provides the management of service usage. By separating access session from service session, a service provider can outsource design, deployment and management of services to third party service providers while still controlling access to these services.

According to these principles, network resources and back-end systems are modeled in a service platform as horizontal platform service components that encapsulate the concrete network technology and topology. Horizontal platform services form a base infrastructure that enables the efficient development of new services and applications. Applications in a service platform shall be independent of the underlying network technology so that a service provider is able to implement services that are accessible to clients using all kinds of access technologies. Access can thus be provided to users for a variety of terminal types, for instance

- Personal computer (PC) with a standard Web browser for rendering HTML/XML/XSL documents, displaying multimedia data with plug-ins, and executing downloadable code such as Java applets.
- Multimedia PCs or smartphones running dedicated applications e.g., a fully featured SIP client.
- Wireless Application Protocol WAP enabled terminals, such as mobile phones and personal digital assistants.
- Mobile phones might also be aware of their current location, e.g., by using the GPS satellite system.
- Plain Old Telephone Sets (POTS) for simple voice communication or interaction using DTMF signaling or voice recognition.

Some services will be dedicated to certain terminal types, but many services will be accessed from more than one terminal type with a varying degree of functionality. These services can share basic infrastructure provided by the SP, shielding the services from the details of the underlying network protocols.

The service platform publishes available platform services to a service registry. Applications act in the role of service requestors that look up available platform services in the registry and finally bind to a selected platform service. An application may thus be composed of a set of platform services and may even become a platform service itself by being published in the platform registry. Service composition is one of the most important features of SOA that enables application design based on the composition and orchestration of generic service components.

According to SOA principles, applications are only loosely coupled to platform service components. From a functional point of view, it thus makes no difference whether an application is developed as part of a network provider's infrastructure, or by a third party. Applications will however run in different security contexts, depending on the level of trust between the network provider and the service provider.

Several common building blocks for developing services for telecommunication networks can be identified:

- User and subscription management, also known as provisioning
- Security management and access control
- Generic session management
- Usage tracking, accounting, billing, monitoring

These common capabilities can be modeled as basic or horizontal platform services, which are always present for all services hosted on the platform.

Apart from basic services, the service platform encapsulates and offers access to service enablers (messaging, location) or back-end systems (data base management systems, rating and billing systems, customer care systems) specific to the telecommunication network.

Furthermore a service platform shall offer a framework for life-cycle management of deployed services. The life cycle of a service involves numerous steps, including design, implementation, testing, deployment, operation, management, maintenance, and removal.

2.2.1 Service Enabler

A service enabler is a resource or function in a telecommunication network that is exposed to service providers in order to implement services. The service is thus enabled to provide added value by combining service logic and network resources. The entity implementing the enabler function is usually encapsulated by an enabler interface. Organizations like the Open Mobile Alliance (OMA) or the 3rd Generation Partnership Project (3GPP) are publishing standardized interfaces for service enabler access.
Typical service enablers are:

- Messaging (SMS, MMS, Wap-Push, e-mail, Instant Messaging)
- Call control
- User interaction – Interactive Voice Response (IVR)
- Call completion (voice mail)
- User status
- User location
- Charging
- Digital Rights Management
- Presence and availability
- Media streaming
- Push to talk Over Cellular (POC)
- Group management
- Content rendering, media stream transcoding, media conferences
- Device capabilities / device configuration

2.2.2 Backend Systems

A service platform supports application development by offering a set of interfaces to backend systems, operational support systems (OSS) and business support systems (BSS). Typical backend systems are:

- Data Base Management Systems (DBMS)
- Billing for recurring and real-time charges
- Rating systems
- Customer care systems
- Accouting and logging systems
- Operations, Administration and Maintenance (OAM) systems

A successful application does not only depend on service logic and available interfaces to telecommunication enablers but also requires integration into a service provider's business processes. A service platform should therefore offer a set of backend service components that offer a useful abstraction of backend and supporting systems.

2.2.3 Parlay

Parlay [Parlay] is a forum of many of the key players in telecommunications and computer industry with the goal to define object oriented APIs that allow third party application devel-

opers to access network resources and functionality in a generic and technology independent way.

In the traditional PSTN systems each Network Provider runs its own service logic on top of his network. In some cases standardized interfaces are used for the connection between the service logic and the underlying network, but in many cases vendor specific products are applied.

This fact leads to the following restrictions:

- Only Network Providers are able to act as Service Provider
- The implemented services are usable only in the Network Provider's network
- Service development is time consuming and costly

This means that companies that want to provide third party services to end-users have three possibilities. First they could restrict their activities only to the IT/Internet domain only, where such restrictions do not exist. Secondly they could run their own network, which would be impossible for the most companies. Finally they could sign a contract with a network provider and implement their services in the network provider's equipment respectively implement some vendor based interfaces, but these services would be applicable only for this one network provider.

On the other hand the Network Providers are very interested in bringing added value to the user as these additional services will increase the usage of the network and value-added services could be charged additionally. In many cases the Network Provider however does not have the knowledge to design and offer these special contents so that the cooperation with a Service Provider is necessary. The Service Provider understands a specific value added service and may even have business connections to his clientele but lacks a network for a large-scale deployment.

Now there are two possibilities to bring added value to the user. One is to use some kind of existing mechanism to bring application-specific data to the user's terminal. The Wireless Application Protocol (WAP), HTTP or any other protocol transporting XML based data can be used as a generic means to control services in the terminal. The terminal however has then to interpret the received scripts and to provide standardized interfaces to its communication resources in order to make converged services possible. Unfortunately this requires not only substantial processing power in the terminal which increases costs and decreases stand-by time for one battery charge in case of a mobile device, but also the interfaces towards the communication resources have to be available in the terminal.

The second possibility to offer value added services to the user is to implement a client-server approach where the terminal is used as a media channel end-point implementing a simple user interface while the service and business logic runs on service platform servers in the network. This is the approach taken by the 3rd Generation Partnership Project (3GPP) by defining the Open Service Architecture (OSA), offering virtual home environment (VHE) type of functions.

The client-server approach uses a standardized interface between the implementation of the Service Logic and the Network Domain. Thus Service Providers would be able to develop services, which are executable on the top of different Network Providers protocol stacks focusing their resources only on the implementation of the service and not on the interoperability to the equipment of different Network Providers. Moreover a meaningful combination of services in different types of networks would be possible. So new and value added services could be developed in less time.

With a standardized API it is possible to combine services in the PSTN, the Mobile Domain and the IP Domain. Furthermore new converged services are imaginable. Some examples could be call specific messaging or multimedia services.
Especially multimedia services seem to be very important for the area of third generation networks because the strongest argument for changing to the third generation mobile networks is the possibility to support more bandwidth consuming applications like video- and audio streaming or other multimedia services.
The architecture shown in Figure 2-1 could overcome these problems.

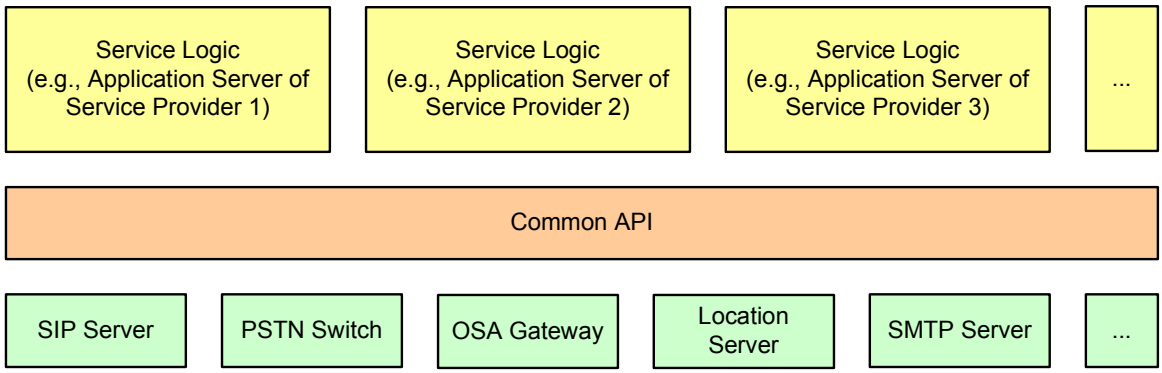


Figure 2-2: Flexible architecture between Service Providers and Network Providers

The Parlay API seems to be a good solution for a common resource API because of the following reasons:

- Many companies of the telecom industry are members of the Parlay group. This promises high acceptance.
- Parlay offers a mature framework for security (authentication, authorisation, service execution safety, high availability), service discovery, service subscription, and service management.
- The IN model is supported by Parlay. Thus backwards compatibility to legacy IN services is possible. Backward compatibility is a very important argument for using the Parlay API as the deployment of third generation networks will take some time and those new networks will co-exist with existing mobile network technologies like GSM or GPRS. CAMEL, the IN version for cellular systems is supported by Parlay.

The Parlay APIs have their origins in the TINA-C specifications [TINA-C] and Intelligent Network (IN) architecture [IN] that is widely used in today's telephony networks. The IN service framework, however, is not an open platform for application development and IN service creation is a time consuming task and restricted to the network provider domain. The Parlay forum learned from the deficiencies of the IN architecture and separated the un-trusted service provider domain from the trusted network provider domain.
Furthermore, Parlay service capabilities are defined far beyond the scope of simple telephony and include the possibility to create powerful applications that combine the strengths of data networks with multimedia communications. A Parlay client application can be developed and run by Third Party Service Providers outside of the Network Providers core domain. The Parlay APIs between those domains manage access to the network resources and are divided in the Framework Interfaces for security, service registration, service discovery, service subscription and management, and the Service Interfaces for the control of specific network ca-

pabilities like multimedia call control, mobility, messaging, QoS management or user interactions. The Parlay APIs are defined in the Unified Modelling Language (UML).
The Parlay Group works closely with ETSI and 3GPP and the Parlay specifications are co-published by all three bodies. Within 3GPP Parlay is part of Open Services Architecture (OSA).
The overall architecture of OSA/Parlay is based on the following structure (Figure 2-3):

1. Components:
   - OSA Framework (authentication, discovery, management)
   - Service Capability Server/Service Capability Feature
   - Application Server
2. OSA API transport bindings:
   - IDL for CORBA
   - WSDL for SOAP/HTTP
   - JAIN SPA for Java RMI
3. Service Capability Features:
   - Call Control
   - User Interaction
   - Mobility
   - Terminal Capabilities
   - Data Session Control
   - Generic Messaging
   - Connectivity Manager
   - Account Management
   - Charging
   - Policy Management
   - Presence and Availability Management
   - Multi-Media Messaging

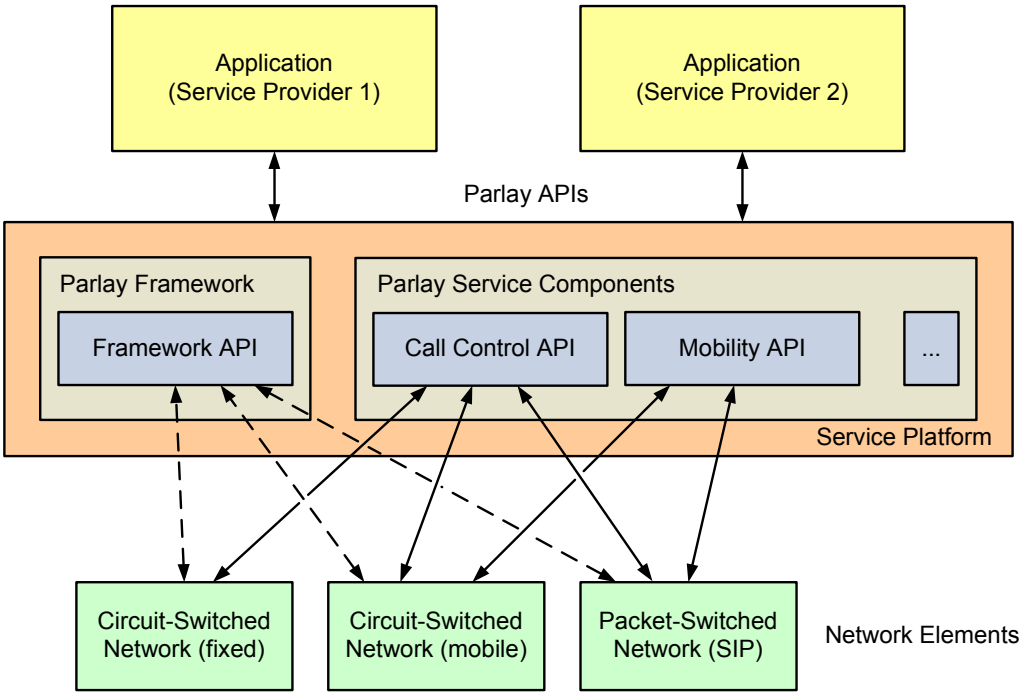


Figure 2-3: Parlay architecture

In 2003 the Parlay Group released a new set of web services called Parlay X [Parlay X]. These are a much simpler set of APIs intended to be used by a larger community of developers. The Parlay X APIs offer Web service specifications for:

- Common
- Third Party Call
- Call Notification
- Short Messaging
- Multimedia Messaging
- Payment
- Account Management
- Terminal Status
- Terminal Location
- Call Handling
- Audio Call
- Multimedia Conference
- Address List Management
- Presence

The Parlay/OSA APIs have an important influence on service platform design, as they offer open and standardized interfaces to many enablers of a telecommunication network. The tight coupling between application and platform by distributed object technologies like CORBA however is opposed to SOA principles. This drawback of Parlay has been mitigated by the publication of the Parlay X Web services.

2.2.4 IMS

While Parlay defines a generic framework that allows third party application developers to access network resources and functionality in a generic and technology independent way, 3GPP's IP Multimedia Subsystem (IMS) specifications [IMS] [Cuevas] is mainly targeted at tird generation wireless UMTS networks and thus (mildly) contradicts the design principle of network independence. The flexibility of the IMS architecture has made it possible however to connect other access networks than UMTS as well. Standardization efforts are under way to integrate additional access networks including legacy GSM networks, cable TV networks, xDSL networks, WLAN networks or WIMAX networks.

The European Telecommunications Standards Institute (ETSI) is currently working on standardization led by the "Telecoms & Internet converged Services & Protocols for Advanced Network" (TISPAN) competence centre for their Next Generation Network (NGN) that aims to define a generalized access architecture for the 3GPP IMS. The NGN specifications have been published in Release 1 [TISPAN NGN] and work for Release 2 is in progress.

[Pavlovski] analyses existing successful service delivery platforms (SDP) and compares those system with the IMS architecture. It is pointed out that many existing SDP deployments are largely IT based designs that are largely associated with user registration, portal design, provisioning, charging and third party service provider integration. It is observed that these IT based designs mainly use Web services as an integration technology. The IMS framework instead very clearly defines the method for integration with network elements and how to manage communications, an aspect that is not well defined in IT based designs.

2.3 Platform Technologies

There is a variety of available platform technologies that are applicable to the design of Next Generation Service Platforms. Platform technologies are used to implement a service platform, but are not necessarily a service platform themselves. Platform technologies are sometimes also referred to as Service Logic Execution Environments (SLEE). Service execution environments offer the following features:

- Environment for service components in order to provide encapsulation and ease re-use of existing code
- Support of various backend interfaces and protocols through generic connectors
- Standardized APIs and architectures provide vendor independence
- A session concept that supports state-full service components
- Container managed persistence (e.g., for writing state to database management system)
- Concurrent execution of service requests supporting threading, queuing, priorities and timeouts
- Scalability by supporting deployment on several parallel hardware platforms
- Transaction handling complying with ACID (Atomicity, Consistency, Isolation, Durability) properties.
- Error handling and fault compensation
- Security
- Monitoring
- Performance (e.g., regarding latency or throughput)

The following lists and evaluates some common platform technologies.

2.3.1 Java Enterprise Edition

Java [Java] is a very popular programming language, in particular for the development of Internet style services. The popularity of Java stems from the operating system and hardware independence due to the interpreting nature of the virtual machine concept and also from the large number of libraries that are available, many of them even in open source license models. Java Platform Enterprise Edition [Java EE] is a technology for developing, deploying and managing multi-tier, server-centric applications. Java EE compliant applications can be deployed on any application server compliant to the Java EE specifications. There are numerous vendors offering Java EE application servers, some of them offering open source licensing models.

Java EE version 5 is an umbrella specification that includes the following APIs:

- Java EE 5
  Java Platform, Enterprise Edition 5 (Java EE 5) JSR 244
- Component Model Technologies
  Enterprise JavaBeans 3.0 JSR 220
  J2EEConnector Architecture 1.5 JSR 112
  JavaServlet 2.5 JSR 154
  JavaServer Faces 1.2 JSR 252
  JavaServer Pages 2.1 JSR 245
  JavaServer Pages Standard Tag Library JSR 52
- Web Services Technologies
  Implementing Enterprise Web Services JSR 109

Java API for XML-Based WS (JAX-WS) 2.0	JSR 224
Java API for XML-Based RPC (JAX-RPC) 1.1	JSR 101
Java Architecture for XML Binding (JAXB) 2.0	JSR 222
SOAP with Attachments API for Java (SAAJ)	JSR 67
Streaming API for XML	JSR 173

- Management Technologies

J2EE Management	JSR 77
J2EE Application Deployment	JSR 88
Java Authorization Contract for Containers	JSR 115

- Other J2EE Technologies

Common Annotations for the Java Platform	JSR 250
Java Transaction API (JTA)	JSR 907
JavaBeans Activation Framework (JAF) 1.1	JSR 925
JavaMail	JSR 919
Web Service Metadata for the Java Platform	JSR 181

2.3.2 JAIN SLEE

The Java API for Integrated Networks Service Logic Execution Environment (JAIN SLEE) 1.0 standard (JSR-22) [JAIN] defines a set of Java technology APIs that enable the rapid development of Java based next generation communications products and services. The Java APIs defined through JSR-22 aim to bring service portability, network independence, and open development to telephony, data and wireless communications networks.

JAIN SLEE is built around a general event model that is based on a publish/subscribe pattern. Event routing between data resources and SLEE components, including inter-component event routing, is a core function of the SLEE. This model decouples event producers from event sources via an indirection mechanism which routes event from sources to consumers, and is referred to as the SLEE event routing mechanism.

The SLEE component model is targeted at event driven applications and thus supports exclusively asynchronous applications. Event producers (e.g., a network resource) and event consumers (e.g., an application component) are only loosely coupled. A SLEE component is called a Service Building Block (SBB) and is hosted by a SLEE container. A SBB complies with certain idioms and programming restrictions. The container acts as the building block's run-time environment and provides services such as resource and life cycle management, concurrency, security, persistence, or transactions. The SLEE container is a specialized light-weight environment for high-speed, low-latency event processing.

The SLEE architecture further defines how applications running within the container interact with resources through resource adaptors. The resource adaptor architecture is important in addressing integration of event driven resources and provides a framework to send and receive events to different network resources.

The SLEE specification also defines a generic provisioned data schema that is easy to use to define, provision, and access profiles. It includes interfaces for adding, removing, and modifying provisioned data. Typical provisioned data includes configuration data, such as per-subscriber data.

A JAIN SLEE is well suited to implement event driven applications like signaling protocol stacks or call state machines. A JAIN SLEE container is able to host a large number of light-

weight and fine-grained objects that have short transient lifetimes and need rapid creation and deletion.
The JAIN SLEE standard however is rather young which leads to the consequence that only a few JAIN SLEE platforms are commercially available and that the developer's community is not very large.

2.3.3 Microsoft .NET

Microsoft's .NET framework is a software component that runs on the Windows operating system. It contains a large class library that covers a broad ranger of application development requirements like data access, web applications, network communications, user interfaces or security. Applications developed for .Net run in an own runtime environment called Common Language Runtime (CLR). The CLR provides the view of a virtual machine to the application and programs are compiles into byte-code that gets executed by a Just-In-Time (JIT) compiler.
Microsoft's C# programming language and the .NET framework are very similar to Java and J2EE. :NET however is currently only fully available on Windows platforms, whereas Java is available on many platforms.
Microsoft's .NET should be considered as a service platform technology, if close interworking and integration with other Microsoft products, like MS Outlook or MS Messenger are required.

2.4 Service Oriented Architecture

Service Oriented Architecture SOA [Zimmermann] is a software design concept that is based on the notion of a service. A service is a software component that a service provider publishes, that has a well defined interface and that is bound and invoked by a service requestor in order to produce some result. The requestor is only loosely coupled to the service, meaning that the interaction is solely based on message exchange specified by the service interface description and does not make any additional assumptions on the details and technology used for the implementation and invocation of the service. Applications built from service components can itself act as a service component again, so that modular building blocks can be arranged in different layers of complexity. SOA thus describes a distributed computing architecture that provides both intra- and cross-enterprise application integration and collaboration.
The main SOA principles can be summed up as follows:

- Services shall be accessible in a platform neutral way
- Service interfaces encapsulate service implementation details
- Service invocation is independent of service location
- Service invocation is message orientated

A more detailed introduction to SOA and Web services is given in Chapter 3 *Service Platform Architecture and Design.*
SOA principles are extremely valuable in solving the complex issues in design and architecture of service platforms for next generation telecommunication networks. It shall be emphasized, that only by structuring a service platform in well defined and loosely coupled service components, the processes of a telecommunications service provider become manageable. The consequences of too complex dependencies between service components usually result in not maintainable application software, unwanted service interactions, intransparent service malfunctions and unpredictable application performance.

2.5 Service Interaction

The execution of several concurrent applications and services for a customer in a service platform context requires coordination between the involved components in order to offer seamlessly integrated services while avoiding negative interactions.
3GPP for example has introduced in [3GPP TS 23.002] a Service Capability Interaction Manager (SCIM) that is part of the application server infrastructure and performs service interaction management (see Figure 2-4). There is however no further specification of SCIM functionality by 3GPP.

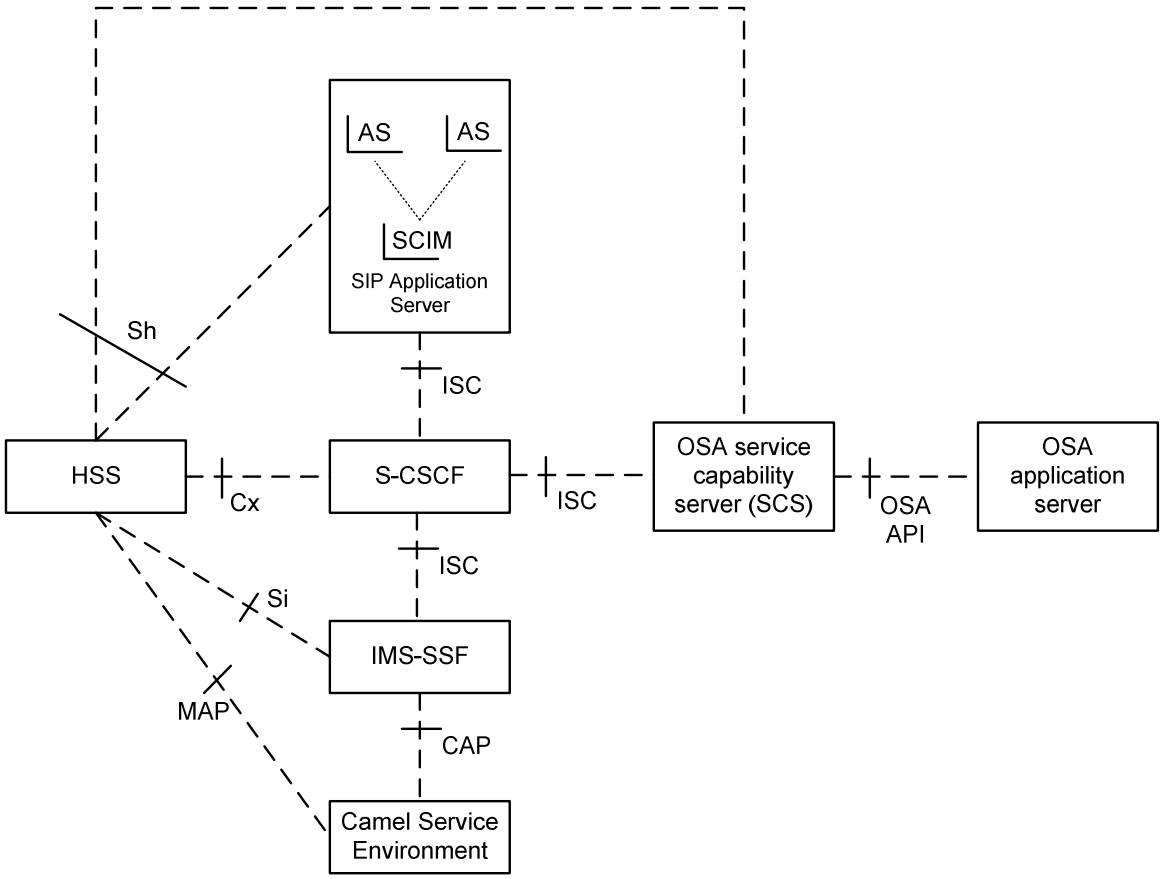


Figure 2-4: Functional architecture for the provision of service in the IMS (from 3GPP TS 23.002)

3GPP's view of the SCIM component may be interpreted in a sense that a SCIM is as a single service interaction layer between the S-CSCF (Serving Call State Control Function) and the application servers that shall be managed. Thus service interaction management would be located in the signalling protocol path of a service platform and is thus depending on the network technology.
This thesis rather proposes a different approach to resolve service interactions that is influenced by the principles of SOA, which is based on the design principle of loosely coupled service components that are invoked by a requestor and offered by a provider. A service component is described by a WSDL document that describes generically the data types, messages and operations of a service and specifically a transport protocol binding (e.g., SOAP/HTTP). Requestor and provider are dynamically bound to each other. A SOA approach for the design of service platform applications offers the important advantage that applications are independent of the network technology.

The SOA approach to service interaction management splits service interaction management in the following layers:

- Business process orchestration:
  Every business process that is independent of the network technology shall be modelled as an executable business process workflow. Service interactions between service components and dependencies of business processes have to be modelled on the orchestration layer. Business processes orchestrate service components.
- Service components:
  Application and service components are implemented as service platform components. Service enablers like call control, media control, interactive voice response, call transfer, call hold, messaging, presence notifications or subscriptions are modelled through protocol independent Web Services (e.g., OSA/Parlay X Web services, OMA Web services). Service components are orchestrated by the business process layer.
- Protocol specific service interactions:
  There may be the need to manage network protocol specific service interactions like e.g., a SIP node that works solely on the SIP protocol layer and extends the core IMS SIP message routing (e.g., for customized header handling, load balancing or monitoring).

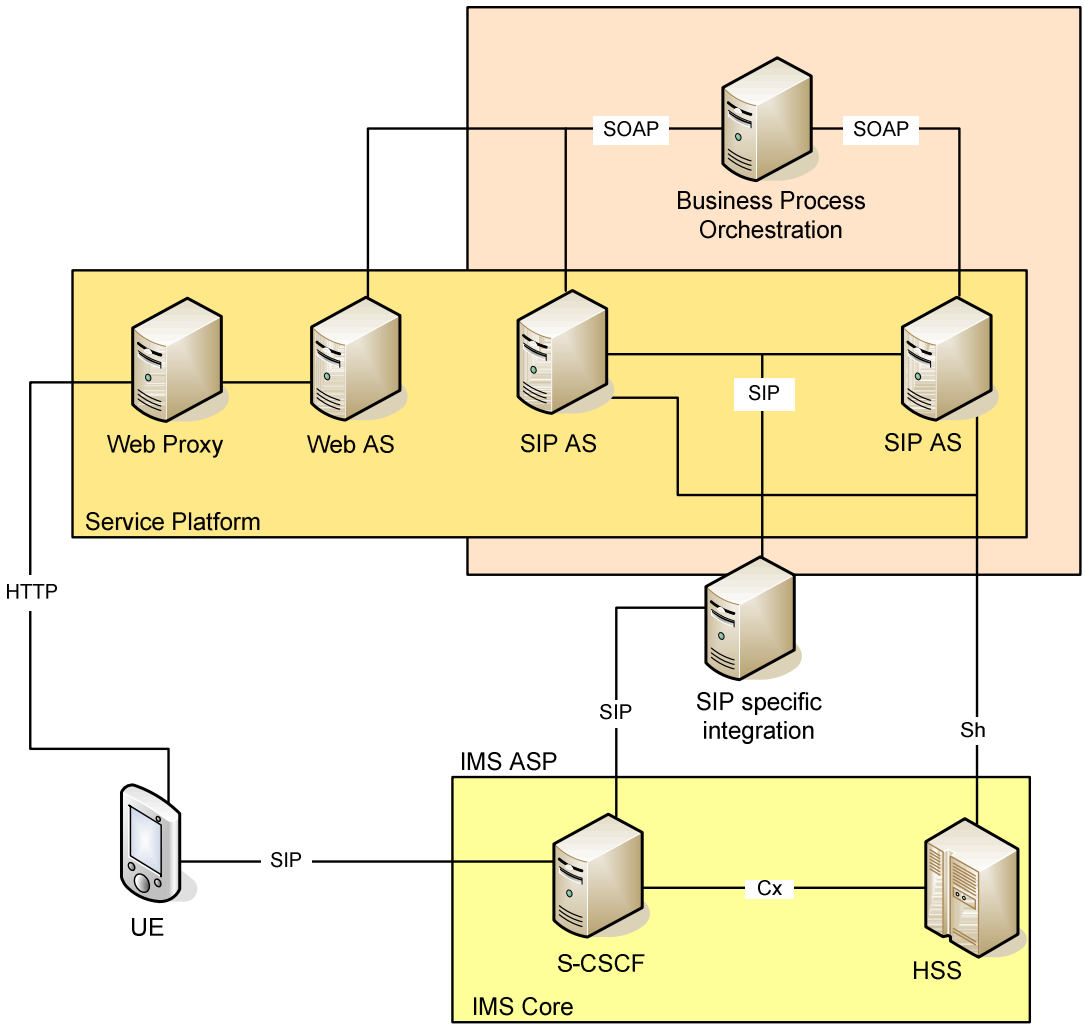


Figure 2-5: SOA based service interaction management

Figure 2-5 shows an example of an IMS service platform, where service components running on different application servers are orchestrated by a business process engine. The service components are implemented as Web services and are independent of the network technology. SIP protocol specific features (if necessary) are handled in dedicated, protocol specific nodes.
It is proposed to see service interaction management as a logical function implemented on different tiers (business process orchestration, web services components, network protocol message processing and routing) and that is distributed between different nodes in those tiers.

2.6 FTW Service Platform

As a concrete example for a service platform for next generation telecommunication network, the FTW service platform shall be presented (see Figure 2-6).

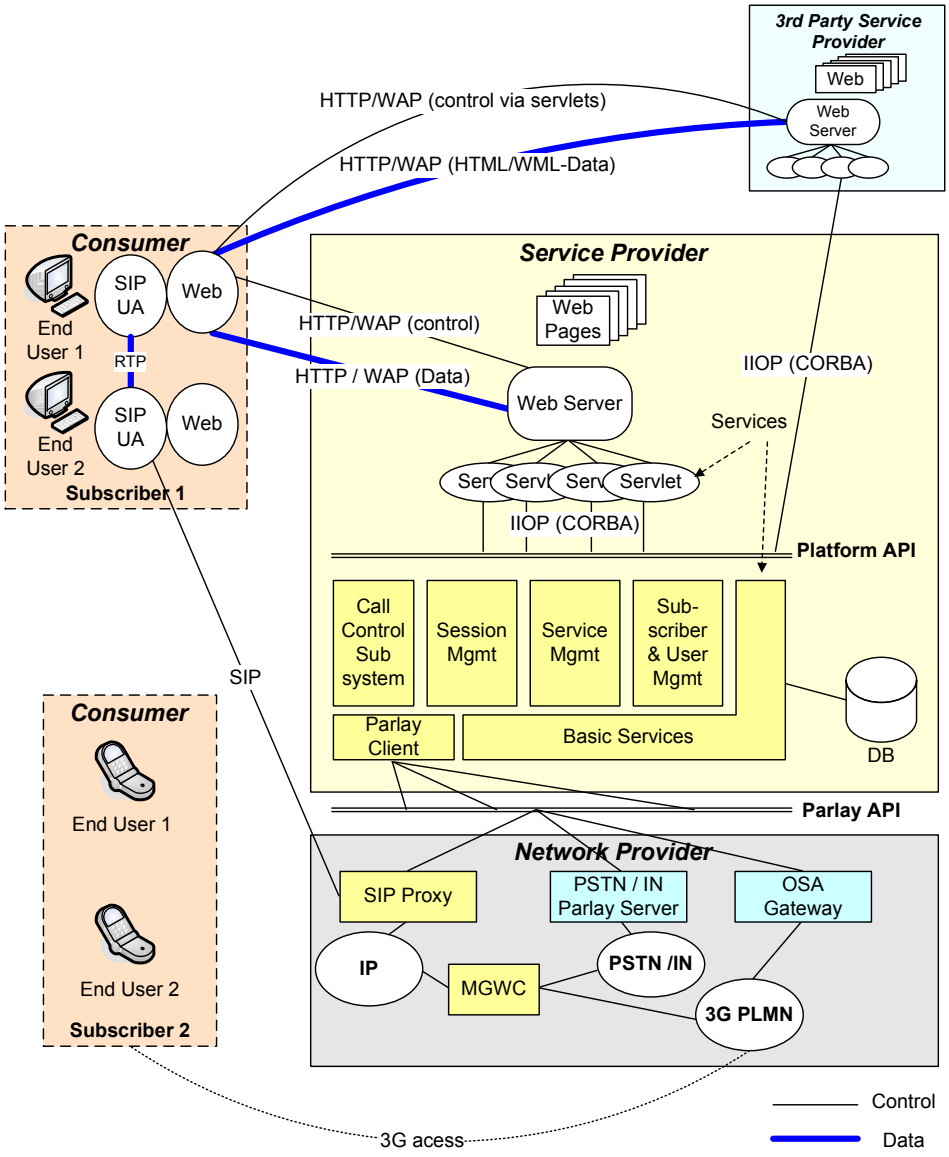


Figure 2-6: FTW service platform architecture

The research work in the projects “A2-Service Platforms and Interoperability” and “P2 – Service Platforms at the Telecommunications Research Center Vienna resulted in a concrete implementation of the proposed service platform concept that was also licensed to a Finnish telecom operator. This implementation is based on the Parlay APIs and CORBA middleware. As discussed in this thesis, the use of CORBA or any other distributed object middleware requires a rather tight coupling of applications with service components and thus contradict the SOA principle of loose coupling. An alternative approach that uses only Web services interfaces will be presented in the next chapter. The Parlay consortium has also acknowledged this shortcoming and as a reaction has provided the Parlay X Web Services and also a Web Service mapping of the original Parlay APIs.

Figure 2-6 illustrates our approach for the realization of the FTW service platform, which integrates different (PSTN, PLMN, IP) networks and services. The FTW Service Platform Solution is based on following principles:

- The customer (end user side) access works via HTTP and WAP. Thus both, the wire-line (LAN, WAN) and wireless mobile (GSM, GPRS and UMTS) terminals are supported.
- The requirements on the end-user-terminals are only that they are able to run a usual Web-browser (HTTP)/WAP and a SIP User Agent.
- The service provider is the platform operator. The service provider owns the service platform and has contracts with third party service providers. He holds all the subscriber data, which is the most valuable asset in the Internet world. This service provider, together with all other contracted third party providers, builds a virtual service domain.
- A central idea is one point authentication in this whole virtual service domain.
- The application / service logic is distributed among the third party service providers. Every third party service provider can provide services independent of other service providers if needed. He needs only a contract with the platform operator if he wants to use services of Service Platform as authentication (one-point-shopping), call control (connection provisioning), media streaming or location services.
- Application / service logic is implemented on top of the HTTP Servlet API and SIP Servlet API.
- Every contracted third Party Service Provider can access the service platform over CORBA interfaces.
- The Service Platform acts as controlling and switching centre for cross domain (PSTN, PLMN, IP) network connections and intelligent communications services such as call forwarding.
- The service platform uses PARLAY as the controlling interface to the underlying networks.
- SIP is used as signalling protocol.
- In accordance to the decision to use PARLAY a SIP-Proxy including a Parlay gateway was implemented.
- Only the control data passes the service platform. The user data is transferred transparently between end-points, that is without any Service Platform interaction. An exception to this point is the IP-PSTN-Gateway, which will be part of the service platform because of the lack of such gateways in the used network.
- SIP is not only used for voice-communication, but for all types of multimedia communication as e.g., video streaming.
- The implementation is based on JAVA, JAVA Media Framework, CORBA, SIP, and the HTTP/SIP Servlet API.

3 Service Platform Architecture and Design

This chapter describes and analyses the required technologies that are necessary to implement service platforms for next generation telecommunication networks. The results presented here are based on research that was carried out within the scope of the projects "A1 - Service Platforms and Interoperability" [FTW A1] and "P2 – Service Platforms" [FTW P2] of the Telecommunications Research Center Vienna (ftw.). In the course of these projects a prototype telecommunications service platform, was developed that is based on the following design choices:

- The service platform is implemented in a standard Java Platform Enterprise Edition (Java EE) application server
- Integration with communication resources is done using the Session Initiation Protocol [SIP]. SIP has been identified as the most promising protocol that combines signaling for multi-media services with a session concept.
- The SIP Servlet API [SIP Servlet] is used as a resource adapter towards the SIP communication network. The SIP Servlet API has been identified as a very useful extension to the Java EE specification that abstracts from the SIP protocol while implementing Java EE programming pattern.
- Service components implemented Web service interfaces that are described by the Web Service Description Language (WSDL).
- The Parlay WSDL interfaces and Parlay X Web service specifications [Parlay] have been used wherever possible to provide service components with open and standardized interfaces. Parlay Call Control interfaces have for example been used to abstract from SIP as the concrete signaling protocol.
- Applications are developed based on network technology independent service components. Interaction and composition of service components are designed according to the principles of a Service Oriented Architecture (SOA).

First this chapter will give an overview on the Session Initiation Protocol (SIP) and the SIP Servlet API. Then a mapping of SIP functionality to Parlay services is proposed [Pailer 01], [Pailer 03] and a prototype implementation using the SIP Servlet API is described. Next an introduction to Service Oriented Architecture (SOA) principles and Web services is given and the advantages and disadvantages of using Web Service technology for service access in telecommunication service platforms are discussed. A prototype implementation of a service platform offering Parlay Web services is presented in detail. Finally softswitch architectures based on the Parlay APIs are presented [Stadler] that implement a telecommunication switching centre as an application running on a service platform.

3.1 Session Initiation Protocol (SIP)

During the projects "A2-Service Platforms and Interoperability", our research group at the Telecommunications Research Center Vienna identified the Session Initiation Protocol (SIP) to be one of the most promising ways for a large-scale deployment of multimedia communications over data networks, in particular the Internet [SIP]. Since then, SIP has been adopted by the 3GPP for their IP Multimedia Subsystem (IMS) and emerged as the preferred Voice over IP (VoIP) protocol.

SIP is an application layer protocol, used for signalling in IP networks. Originally it was developed by the Multiparty Multimedia Session Control (MMUSIC) working group of the IETF for signalling of large multicast conferences. The scope has since then shifted towards signalling of two-party calls, but of course multiparty conferences are also supported. SIP allows establishment, modification and termination of all types of sessions. It is a text-based, HTTP-like peer-to-peer protocol with two basic types of messages, requests and responses. SIP messages carry information about communication sessions. For VoIP applications and multimedia conferencing applications this information is usually placed in the Session Description Protocol [SDP]. In that case the SDP packet is a part of the SIP message, called body. Another important IETF protocol for VoIP is the Real-time Transport Protocol [RTP]. It is used for the transport of real-time media data (voice and video) in an established session.

SIP defines two entities that build an overlay network, the SIP user Agent (UA) and the SIP network server. A SIP UA could be seen as end device and acts either as user terminal or as automated connection endpoint, for instance a call answering machine. The SIP UA implements the functions of a UA Server and a UA client. UA clients initiate calls or more generally send requests, while UA servers respond to requests (e.g., a call invitation). The UA is a state-full entity. This means that call state is held in the call end-point.

SIP network servers are used for call routing and for maintaining call states, as well as they could be enabled to perform different kinds of applications. They are divided into three different types, the proxy server, the redirect server and the register server.

A SIP proxy server is an application layer router that forwards SIP messages after address resolution. SIP redirect servers reply to incoming SIP request with a new user location. After receiving a response from a SIP redirect server, a SIP UA will try to reach the called party at the new location.

SIP message routing is one of the most important functions of a SIP network server. Because of scalability reasons, SIP messages carry all necessary routing information so that SIP servers can be stateless. There may however be circumstances (e.g., in an edge proxy) where SIP servers have to be state-full to provide all required functionality.

Finally there is the SIP registrar server, where UA register with the current network address (e.g., a SIP URI) of the terminal. The user's current network address is stored by the registrar server. After registration the register server is able to find out the current SIP address on the basis of the unique SIP address. Validity of a register message is always time limited. Users can also un-register by sending a register message with a time limit of zero. The register mechanism in SIP ensures that a user is reachable anywhere having only one unique SIP address. This mechanism is also used for mobility support in SIP and thus SIP is a location independent protocol that does not bind a user to a specific terminal or network.

The session concept of SIP offers a generic way to establish, control and tear down multimedia sessions between communication end-points. It is one of the strengths of SIP that is does

not only enable VoIP services, but also supports the set-up of multimedia streaming sessions as well as interactive multimedia communications.
SIP messages are divided into requests and responses. A very important part of a SIP request is the method, which specifies the type of request. The [SIP] defines six methods: INVITE, ACK, BYE, CANCEL, OPTIONS and REGISTER. SIP responses are specified by the status code and the reason phrase. They are divided into provisional, successful, redirection, request failure, server failure and global failure responses. There is only one successful response defined that is the 200 OK response.
A SIP message can also carry a description of the session in its body. Usually this is the case for an INVITE request and a 200 OK response to an INVITE request. For "voice over IP" (VoIP) applications the session is mostly described by the Session Description Protocol [SDP]. Another important IETF protocol for VoIP applications is the Realtime Transport Protocol [RTP]. It is used for the transport of real-time media data (voice and video) in an established session.
Session setup in SIP is quite simple (see Figure 3-1). Suppose there are two users (A and B) and user A wants to set up a call to user B. User A sends an INVITE request to B. If B wants to accept the call, he replies with OK response. The last step is the sending of an acknowledge (ACK) request by A, after the response arrives from B. Afterwards, the media data (e.g., video and audio) can be transmitted using RTP. For reliability reasons, SIP uses a three-way handshake (INVITE, OK, ACK) for the call setup. If one party wants to tear down the call, it simply sends a BYE request to the opponent UA, which replies with an OK response and the call is terminated.

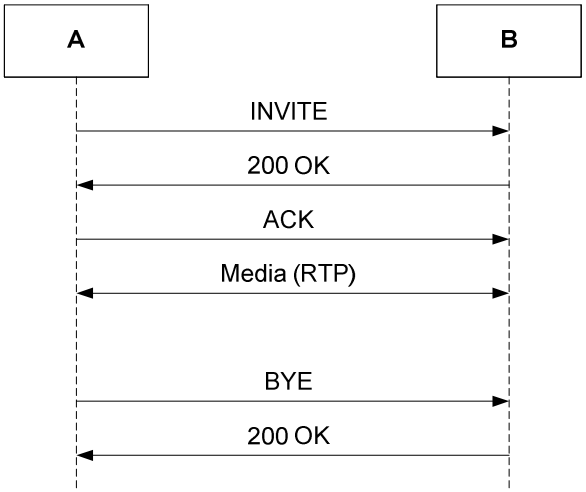


Figure 3-1: Basic SIP call

SIP is also an extendable protocol. There are different SIP extensions that make possible realisation of advanced services in SIP. Especially interesting are extensions for event notification [RFC 3265], presence [RFC 3856] and instant messaging (IM) [SIMPLE] [RFC 2778]. These specifications define two new methods, SUBSCRIBE and NOTIFY, for presence and a new method MESSAGE for IM.

3.2 SIP Servlet API

During the projects "A2-Service Platforms and Interoperability" and "P2 – Service Platforms", our research group at the Telecommunications Research Center Vienna [FTW] was actively involved in the Java Community Process for the Java Specification Request 116 "SIP Servlet API" [SIP Servlet] [Peterbauer]. To the author's knowledge our project was the first group to implement and license a SIP server based on an implementation of the SIP Servlet API.

Meanwhile the SIP Servlet API has developed into the de-facto standard for SIP service development in Java Enterprise Edition environments. This section gives an overview on the SIP Servlet API specification.

SIP servlets are Java based service components that implement SIP signalling and run in a SIP servlet container, sometimes also called servlet engine. SIP servlets use the SIP Servlet API to send SIP request messages and to receive SIP response messages. The SIP Servlet API is thus a standardized way to access the SIP protocol stack and to interact with other SIP nodes.

JSR-116 does not only specify a pure SIP protocol API, but also defines the SIP servlet container, that runs in a application server and is responsible for SIP application initialization by deployment descriptors and life-cycle management of SIP servlets. The container is responsible for creating and destroying SIP servlets and decides which SIP servlets to invoke and in what order.

The SIP servlet API builds on the very successful HTTP Servlet API [Servlet] and in fact the specification encourages that SIP sessions and HTTP sessions may be correlated in one application session of a converged container. There are however important differences between the SIP and HTTP protocols. HTTP is a pure client-server protocol where the initial request always originates at the client and the final response is always generated by the Web server. SIP on the other hand is a peer-to-peer protocol where a user agent implements a client and a server. Intermediary SIP server nodes may proxy and fork requests, may generate multiple responses, may receive responses as well as requests and may even initiate requests. Thus the SIP Servlet API extends the capabilities of the HTTP Servlet API substantially.

3.2.1 Converged Container Use Case

The following use case from the SIP Servlet API specification illustrates the potential of SIP servlets for providing converged applications that blend a Web server with a SIP servlet engine. The application implements the service 'Call Back On Busy'. Alice calls Bob while Bob is busy. Bob redirects Alice to an HTTP URL that offers an Bob's individual 'Call-Back' Web page. Alice selects to be called back as soon as Bob is available again. The application subscribes to Bob's online status and establishes a third party call between Alice an Bob, when Bob has finished the previous call. The message sequence diagram is pictured in Figure 3-2.

A converged container offers the possibility to hold the SIP session and the HTTP session together in one session context. In this example an application session holds context about several SIP sessions and one HTTP session.

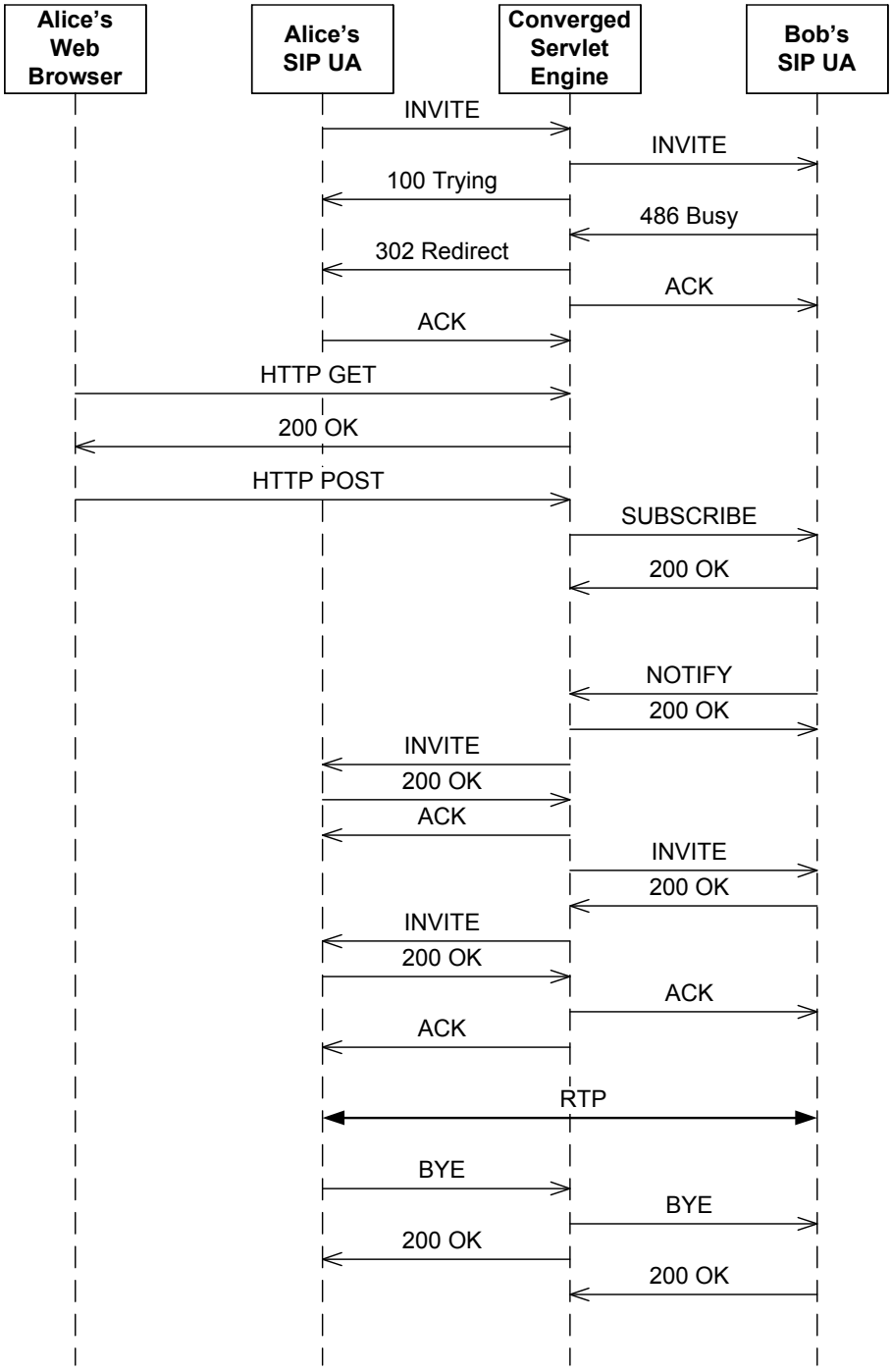


Figure 3-2: Converged servlet engine message sequence diagram

3.2.2 SIP Messaging

The SIP servlet container provides an API to the SIP protocol stack. The container is responsible for:

- Parsing SIP messages
- Delivering SIP messages as *SipServletRequest* or *SipServletResponse* objects to the appropriate SIP servlet
- Forwarding SIP messages generated by a SIP servlet according to SIP routing rules
- Automatically generating SIP messages

- Depending on the role a SIP servlet is acting in (User Agent Client, User Agent Server, Proxy, Back-to-Back User Agent), the SIP servlet container automatically generates requests and responses that are of no interest to the application logic (e.g., 100 Trying response or retransmissions).

- Automatically handling SIP message headers.
  - System headers are those headers that are managed by the servlet container and which servlets must not attempt to modify directly via calls to setHeader or addHeader. This includes the headers Call-ID, From, To, CSeq, Via, Record-Route, Route, as well as Contact when used to confer a session signalling address.

3.2.3 SIP Request Routing

One of the main goals of the SIP protocol is to define message routing rules for a SIP overlay network. SIP applications are executed inside a servlet engine and consist of one or more SIP servlets. Routing of SIP messages towards SIP applications is done as if each SIP application were an isolated SIP node. This follows from the 'cascading services model' specified for the SIP servlet container that is "triggering of service applications on the same host, shall be performed in the same sequence as if triggering had occurred on different hosts".
The container defines routing rules of SIP messages to these applications that specify which applications are called and in what order. The SIP Servlet API specification defines an XML document that contains servlet mapping triggering conditions for the invocation of particular servlets. Given a certain initial request, a triggering rule evaluates to true or false. If more than one rule matches, the corresponding servlets are triggered in the order the matching rules are listed. Rules can define logical combinations of the boolean operators *equa*l, *exists*, *contains*, *subdomain-of* on the contents of the SIP method, the request URI, the To header and the From header. The rules XML is part of the SIP servlet deployment descriptor file *sip.xml*. An example of a servlet mapping is shown in Figure 3-3.

```
<servlet-mapping>
  <servlet-name>messageServlet</servlet-name>
  <pattern>
    <equal>
      <var>request.method</var>
      <value>MESSAGE</value>
    </equal>
  </pattern>
</servlet-mapping>
```

Figure 3-3: Example of a <servlet-mapping> element in a sip.xml file

Triggering rules only apply to initial requests, that is requests that do not belong to any SIP session. It is the task of an application to establish a SIP session for an initial requests which belongs to an application session. A *SipSession* object corresponds to a SIP dialog. An *ApplicationSession* object may hold more than one SIP sessions (e.g., in case the applications implements a B2BUA). When creating a SIP session, the application may record-route, meaning that it can specify if subsequent requests shall be routed to that particular application. Subsequent requests are requests that are sent inside an existing SIP dialog. Subsequent requests are routed to the application that is associated with a particular SIP session and application session, if the session has been established as record-routing.

Subsequent requests from the originator traverse all record-routing applications in the same order as the initial request. Subsequent requests from a terminator traverse all record-routing applications in reverse order of the initial request.
Reponses always traverse applications in the reverse order of the initial requests, independent of record-routing. This behaviour is consistent with SIP response routing according to Via headers.

3.2.3.1 SIP dialogs

SipSession objects either represent an established SIP dialog or the state before the SIP dialog is actually established. The *SipSession* thus extends the SIP dialog state machine by an additional INITIAL state. Figure 3-4 shows the SIP dialog and the *SipSession* state machines.

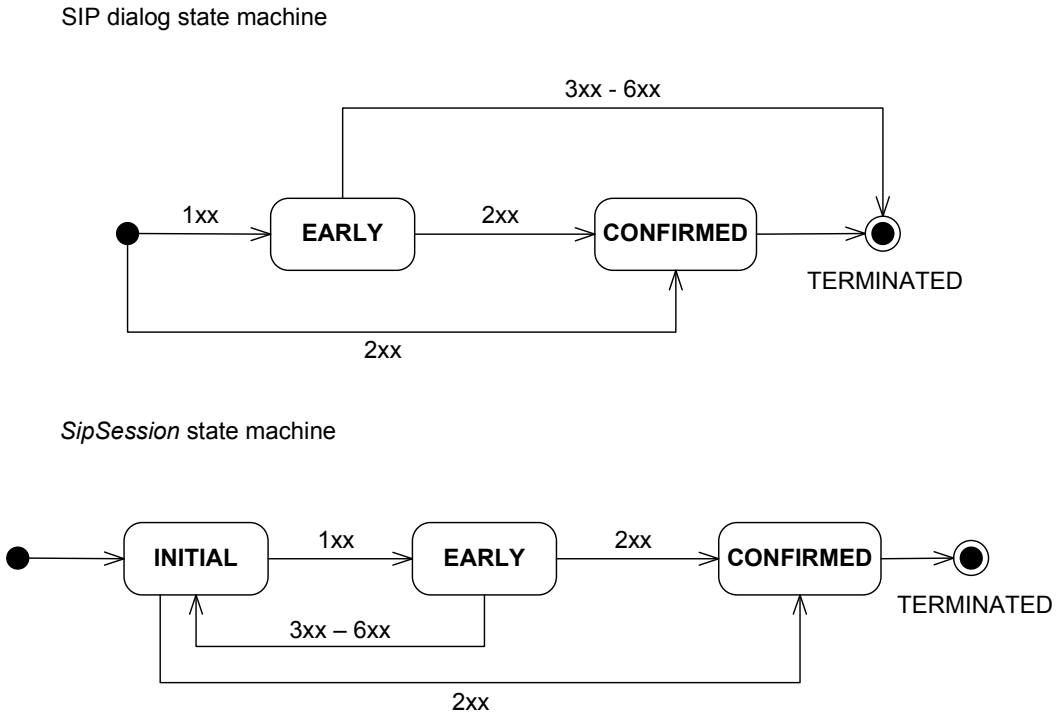


Figure 3-4: SIP dialog and SipSession state machines

The *SipSession* state machines differs from the SIP dialog state machine in the following ways:

- The INITAL state is added as a valid state. In this state multiple requests can be generated.
- A non-2xx state received in the EARLY state does not terminate the *SipSession*, but triggers a state change back to the INITIAL state, where new requests can be generated.

A big difference between dialogs and *SipSessions* is that whereas SIP requests may exist outside of a dialog (for example, OPTIONS and REGISTER), in the SIP Servlet API all messages belong to a *SipSession*.

3.2.4 SIP Servlet

The *SipServlet* interface is the central abstraction of the SIP Servlet API. The servlet lifecycle is controlled by the container. The lifecycle state diagram is shown in Figure 3-5. The servlet container loads the servlet class, instantiates it and invokes *init()*. Now the servlet is able to process SIP messages in the *service()* method. When the servlet container decides to dispose

of the servlet, the *destroy()* method is invoked to free all resources allocated in *init()* and afterwards the servlet instance can be garbage collected.

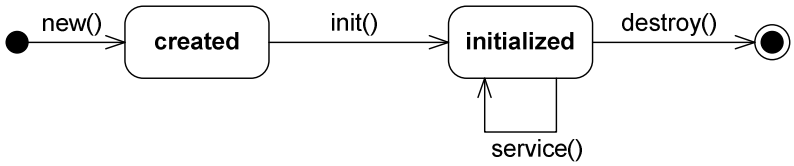


Figure 3-5: SIP servlet lifecycle

SIP message processing is handled in the *service()* method. SIP messages can be dispatched concurrently by the container to a servlet, so that servlets have to be implemented in a thread save manner. The SipServlet implementation of the *service()* method dispatches incoming messages to the *doRequest()* and *doResponse()* methods for SIP requests and SIP responses, respectively.

The *doRequest()* implementation of the *SipServlet* passes requests further on to the following methods:

- *doInvite* for handling SIP INVITE requests
- *doAck* for handling SIP ACK requests
- *doOptions* for handling SIP OPTIONS requests
- *doBye* for handling SIP BYE requests
- *doCancel* for handling SIP CANCEL requests
- *doRegister* for handling SIP REGISTER requests
- *doPrack* for handling SIP PRACK requests
- *doSubscribe* for handling SIP SUBSCRIBE requests
- *doNotify* for handling SIP NOTIFY requests
- *doMessage* for handling SIP MESSAGE requests
- *doInfo* for handling SIP INFO requests

The first six SIP messages are specified in the base SIP RFC 3261 [SIP]. The other methods are defined in various SIP extensions. The PRACK message is specified for the reliable handling of provisional responses [RFC 3262]. The SUBSCRIBE and NOTIFY requests are specified in the SIP event notification framework [RFC 3265] and used in the SIP presence architecture [SIMPLE]. For instant messaging MESSAGE is specified in [RFC 3428]. The INFO message is a general purpose message that can be used inside an existing dialog [RFC 2976].

The *doResponse()* method implementation of the *SipServlet* passes responses further on to the following methods:

- *doProvisionalResponse* for handling SIP 1xx informational responses
- *doSuccessResponse* for handling SIP 2xx success responses
- *doRedirectResponse* for handling SIP 3xx redirection responses
- *doErrorResponse* for handling SIP 4xx client error, 5xx server error, and 6xx global failure responses

Figure 3-6 shows an example of a simple SIP servlet that replys with a 200 OK when receiving an INVITE request.

```
public class ExampleSIPServlet extends SipServlet {

      protected void doRequest(SipServletRequest req)
      throws ServletException, IOException {

            SipServletResponse res = req.createResponse(200);
            res.send();
      }
}
```

Figure 3-6: Simple example of a SIP servlet class

The relationship between the HTTP Servlet API and the SIP Servlet API also shows in the Java package dependencies *javax.servlet* and the *javax.servlet.sip*. The *SipServlet* class extends the *GenericServlet* class from the *javax.servlet* package. The *SipServletRequest* and *SipServletResponse* interfaces extend the *ServletRequest* and *ServletResponse* interfaces from *javax.servlet* respectively. SIP request and response interfaces both extend the *SipServletMessage* interface that defines a number of methods that are common to both requests and responses like:

- SIP header manipulation
- Get the associated application session
- Get the associated SIP session
- Manipulation of the message body content
- Sending this message
- Get the network transport protocol
- Get local and remote network addresses

SipServletMessage objects always implicitly belong to a SIP transaction (see chapter 17 'Transactions' in [SIP]) and the SIP transaction state machine that constrains what messages can legally be sent at various points of processing. The SIP servlet engine has to enforce that only messages compliant to the SIP transaction state machine are sent by a servlet.
Figure 3-7 shows the basic class hierarchy of the SIP Servlet API.

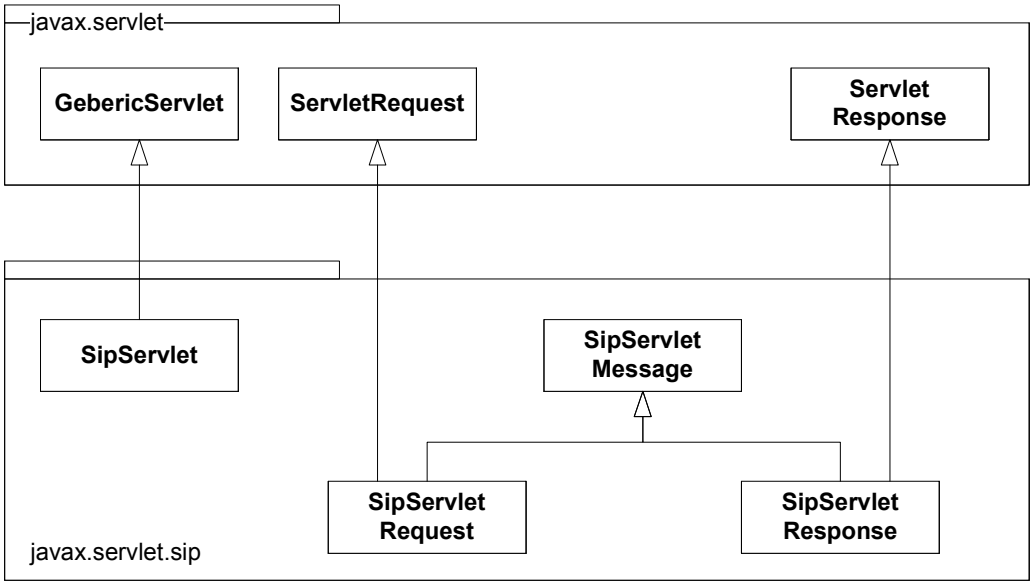


Figure 3-7: SIP Servlet API class hirarchy

3.2.5 Servlet Context

SIP servlets run in a servlet context that is associated with a particular application. A servlet context in a converged container can contain multiple SIP and HTTP servlets (see Figure 3-8).

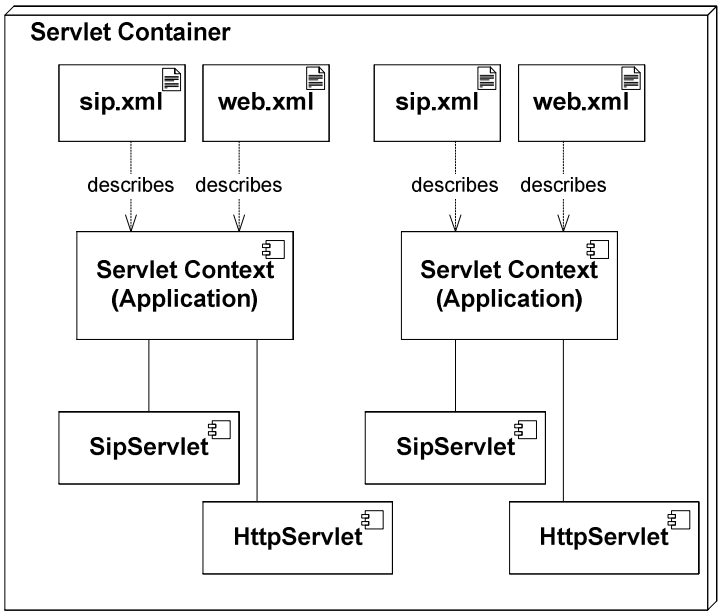


Figure 3-8: SIP servlet container deployment diagram

A SIP application needs a SIP specific deployment descriptor (sip.xml) that is very similar to an HTTP deployment descriptor (web.xml) except for the <*servlet-mapping*> element. In HTTP a servlet is associated with a part of the URL while in SIP a servlet is chosed according to filter patterns.

Configuration and executable files of SIP/HTTP applications are stored in predefined structure and packaged in compressed form in a Web Archive (WAR) file.

The servlet context is created when a SIP application is initialized and provides an implementation of the *SipFactory* interface as a context parameter. The SIP factory has to be used to create *SipApplicationSession* objects on initialization and to create initial requests for a new dialog, when a servlet is acting as a UAC. Requests that belong to an already existing dialog have to be created with the *SipSession* object. The factory creates address objects in form of URIs. SIP URIs and also tel URLs [RFC 2806] are mandarory.

3.2.6 SIP Servlet Roles

A SIP servlet can act in the roles of a User Agent Client (UAC), User Agent Server (UAS), Proxy or B2BUA (Back-to-Back User Agent). The SIP servlet container supports these modes by encapsulating SIP protocol details like retransmissions, automatic handling of particular requests or responses or generation of session tags and IDs. The next sections will give an overview how SIP servlets can implement different SIP roles.

3.2.6.1 Servlet acting as User Agent Client

A SIP servlet acting as a UAC can create initial and subsequent requests and receives responses to sent requests.

Application sessions, SIP URIs and initial requests are created by the *SipFactory* object, that can be obtained from the *ServletContext* object:

```
SipServletRequest createRequest(
```

```
SipApplicationSession appSession,
String method,
URI from,
URI to);
```

The created request belongs to a new SIP session that is associated with the given application session. The created SIP session belongs to the default servlet of the application, but the handling servlet of a SIP session can be changed by the application by invoking `SipSession.setHandler`. The container is required to add a new globally unique *Call-ID* header and a *Cseq* header to the request. A tag parameter is added to the *From* header. The request URI is per default set with the value of the *To* header.

The request headers and content can now be modified by the servlet. and finally be sent by invoking the *send()* method. The servlet container is responsible for automatically generating the system headers *Via* and *Contact*.

If a dialog is established, the tag parameter of the *To* header and the route of the SIP dialog is updated by the container in the SIP session. All subsequent requests belonging to this SIP session will have the same *From* and *To* headers. Subsequent requests can be obtained from the *SipSession* object:

```
SipServletRequest createRequest(String method);
```

The container provides this new subsequent request with values for the *Call-ID*, *CSeq*, *From*, *To* and *Route* headers.

The handling servlet of a SIP session is invoked for all reponses of sent requests except for *100 Trying* and retransmissions. Due to forking proxies in the request path, it may be possible that multiple 2xx responses are received on a single initial request. The servlet container will created a cloned SIP session for each received 2xx response with different tag parameters in the *To* header.

It is the responsibility of the application to generate an ACK request to a 2xx response by invoking the SipServletResponse method:

```
SipServletRequest createAck();
```

The application may modify the content of the ACK request before sending it.

ACK request to reponses other than 2xx are only needed for reliability purposes and are thus generated automatically by the servlet container.

A UAC may cancel an INVITE request in progress by invoking on the original INVITE request object:

```
SipServletRequest createCancel();
```

Reponses to the CANCEL request are not forwarded by the servlet container to the application.

3.2.6.2 Servlet acting as User Agent Server

A SIP servlet that responds to an incoming request with a final response becomes a UAS for this transaction. An application may generate a number of provisional responses before sending a single final response. Reponses are obtained from the *SipServletRequet* object by invoking `createResponse`. The response object may be modified before being sent.

The servlet container is responsible for generating retransmission of responses, also for 2xx responses.

SIP servlets are invoked for ACK requests received on 2xx responses. ACKs to responses other than 2xx are only sent for reliability purposes and are not forwarded by the servlet container to the application.

When a CANCEL request is received and the application has not yet sent a final response, the servlet container automatically sends a *487 Request Terminated* to the original request and a *200 OK* to the CANCEL. Afterwards the CANCEL request is forwarded to the application. The application should not attempt to answer the original request after receiving a CANCEL, nor should it respond to the CANCEL.

3.2.6.3 Servlet acting as Back-To-Back User Agent

A B2BUA is a SIP node that binds together two or more SIP dialogs and forwards requests and responses between those dialogs in some fashion. A B2BUA mediates between two (or more) endpoints, but decouples direct SIP signalling. A B2BUA may potentially break a service between two endpoints that it is not aware of. B2BUA are however very useful, when essential parts of the SIP message have to be changed like an SDP body or Call-ID, To or From headers . Another useful example is a prepaid servlet that sends a BYE request to all endpoints when the prepaid account has been used up.
Generally speaking, a B2BUA is an intermediate SIP hop that exceeds the functionality of a SIP proxy by (sometimes) acting like a User Agent.
The SIP Servlet API supports the implementation of B2BUA by providing a method to create SIP requests, tailored to the needs of a B2BUA while minimizing the risk of breaking end-to-end services. A SIP servlet acting as a B2BUA may invoke on the *SipFactory* object

```
SipServletRequest createRequest(
     SipServletRequest origRequest,
     boolean sameCallId);
```

This method creates a request in a new SIP session that is identical to the one provided, including all headers, with the exception that

- The *From* header field of the new request has a new tag chosen by the container.
- The *To* header field of the new request has no tag.
- A new *Call-ID* is created if *sameCallId* is false.
- *Record-Route* and *Via* header fields are not copied, but are created by the container automatically.
- For non-REGISTER requests, the Contact header field is not copied but is populated by the container as usual.

Note in particular, that the Route header is copied from the original request to the new request.

3.2.6.4 Servlet acting as Proxy

One of the most important task of SIP is request routing, the ability to decide which destination or destinations should receive the request. A node routing SIP messages is called a SIP proxy.
A SIP servlet controls proxying of request via the *Proxy* object. A *Proxy* object is obtained by invoking `SipServletMessage.getProxy()`. If the proxy is transaction stateful, the same proxy instance is returned for one SIP transaction. The following parameters control the proxy behaviour:

recurse: flag specifying whether the servlet engine will automatically recurse or not. If recursion is enabled the servlet engine will automatically attempt to proxy to contact addresses received in redirect (3xx) responses. The default value is true.

recordRoute: flag controlling whether the application stays on the signaling path for this dialog or not. This should be set to true if the application wishes to see subsequent requests belonging to the dialog. The default value is false.

parallel: flag specifying whether to proxy to multiple destinations in parallel (true) or sequentially(false). In the case of parallel search, the server may proxy the request to multiple destinations without waiting for final responses to previous requests. The default value is true.

stateful: whether the proxying operation should be transaction stateless or stateful. The default value is true, meaning stateful. The ability to proxy without maintaining transaction state potentially yields better performance.

supervised: whether the servlet is invoked to handle responses. Note that setting the supervised flag to false affects only the transaction to which the Proxy object relates. The default value is true.

sequentialSearchTimeout: the time the container waits for a final response before it cancels the branch and proxies to the next destination in the target set. The default value is the value of the sequential-search-timeout element of the deployment descriptor or a container specific value, if this is not specified.

If one of the proxy parameters *recurse*, *parallel*, or *supervised* is set to true, proxying cannot be stateless.

An application starts proxying by obtaining a Proxy object from a request. It may then modify the request by adding, changing or deleting headers and content of the request. The application then invokes the `proxyTo` method on the *Proxy* object that takes one or more URIs as destinations. For each URI a new branch is created. The container then routes the request according to the request URI, unless a *Route* header is present. An application may set one or more entries in the *Route* header, by invoking `pushRoute(SipURI uri)` on the *SipServletRequest* object.

An application may generate informational responses before or during proxying operation in the same way as a UAS. The application may also send final responses after proxying a request:

- A 2xx response is sent upstream.
- A non 2xx response created by the application while having outstanding branches is sent to the `doResponse()` method of the same application. Thus a non 2xx response is handled like any other response from an outstanding branch. If it's eventually selected as the best response, the container will perform the usual best-response callback.
- If the best response received was a non-2xx and the application generated its own final response in the `doResponse` callback (be it a 2xx or non-2xx), then that response is sent immediately without invoking the application again for its own generated response. This allows applications to create, for example, a different error response from the one chosen by the container as the best response.

A proxy servlet will only receive responses, if it acts as a stateful proxy. When in stateful mode, the servlet container is responsible for automatically forwarding the following responses received for a proxying operation upstream:

- All informational responses other than 100
- The *best* response received when final responses have been received from all destinations.
- All 2xx responses

Additionally, if the *supervised* flag is true, the servlet engine invokes the application for these responses before forwarding them upstream. The application may modify responses in the

notification callback. This is useful, for example, for application level gateways that need to modify addresses in the body of responses.

If a final 2xx or 6xx response is received, a CANCEL request is sent by the container to all outstanding branches.

If the application is invoked with a non 2xx final response, if may decide to proxy a request to further destinations by invoking `Proxy.proxyTo`. This creates a new branch, that is handled by the container like any other outstanding branch.

The application can correlate between a received response and a branch by invoking `SipServletResponse.getRequest()` on the reponse object. The application can then compare get the request URI from this request, which is identical to the URI that was used as parameter in the `proxyTo` method, when creating this branch.

The application can cancel a proxied INVITE request by invoking `Proxy.cancel()`. The container then sends a cancel to all outstanding branches. Reponses that result from cancelling a branch a forwarded to the application as described above.

When receiving a CANCEL request the container responds with a *200 OK* and

- Responds with a final 487, if the request was not yet proxied
- Cancels all outstanding branches otherwise.

The application is notified of the CANCEL in any case, but should not react on it.
The servlet container is responsible for generating ACKs to non-2xx final responses. Servlets should therefore never generate ACK requests.
When an application is proxying with *recordRoute* set to true, the application is invoked for subsequent requests and ACKs to 2xx responses. The servlet can check, if the request is a subsequent request by invoking `isInitial` on the request object. The application may choose to modify the subsequent request but should not proxy the request explicitly. Proxying of subsequent requests is done by the container automatically. The application may however send a final response to a subsequent request.

3.2.7 Servlet Timer Service

The timer service is a container-provided service that allows applications to schedule timers and to receive notifications when timers expire. Timers can be scheduled to expire once after a specified time, or repeatedly at specified intervals. The timer service has particular importance for the design ob robust SIP applications.

3.2.8 SIP Servlet API Alternatives

Apart from the SIP Servlet APIs, there exist several alternative technologies that could be used as an API towards a SIP protocol stack. In the following the issues with some of these technologies are listed:

- SIP Common Gateway Interface (CGI):
  SIP CGI [RFC 3050] is very similar to HTTP-CGI except some little adaptations, e.g., persistence issues. Therefore it takes on most of the problems of this mechanism. Performance and portability depend on the language behind the CGI interface. Because the information exchange is done by environment variables it has to run on the same system like the server. This could be a problem for scalability. Furthermore all needed environment variables have to be parsed. This is not very convinient for the application

programmer. Moreover, as can be seen from the name, it is only applicable for SIP Servers. A combination of services between different network technologies is not possible.

- Call Processing Language (CPL):
  CPL [RFC 3880] is an XML derivate script language and is very useful for user controlled call routing. CPL is very promising in the service domain, which means above the Parlay API. Besides the advantage of defining services for different network technologies (maybe here some adaptations may be necessary, as CPL is very close to SIP functionality) the CPL functionality would be complemented by the service framework of Parlay regarding management and security. Furthermore CPL is not Turing-complete, which implicates that it provides no way to write loops or recursion. So there are some limitations in the functionality of such scripts that certainly are useful to ensure safety of scripts designed by the end-user but may be too restrictive for carrier-grade service design. Finally CPL does not offer the possibility to trigger a third party call set-up, which is of crucial importance for many third party applications.
- Java Advanced Intelligent Network (JAIN):
  JAIN [JAIN] offers the possibility to use implemented services in different networks. Like Parlay it abstracts from the used network protocol stack. Furthermore a SIP API has beenspecified [JAIN SIP]. The JAIN SIP API is much more detailed than the SIP Servlet API. The application developer has to deal with more details of the SIP message processing than is necessary for establishing a generic call control model. As the name says JAIN is very Java dependent and strongly connected to the Enterprise Java Bean (EJB) concept. JAIN system design is based on an event-listener framework and resembles in that respect the programming pattern of classic telephony switches. JAIN offers a network independent platform for service development.

3.3 Mapping SIP to Parlay

In this section an architectural solution is proposed that specifies how the interworking between SIP and Parlay can be implemented and shows this interoperability by means of some scenarios for multimedia communications.

In 3GPP's service framework the use of the Parlay APIs is proposed that allow application development by third parties in order to speed up service creation and deployment. 3GPP has also adopted SIP for session control of multimedia communications in an IP network. This section proposes a mapping of SIP functionality to Parlay services and describes a prototype implementation using the SIP Servlet API. The presented mapping was first proposed in [Pailer 01], [Pailer 03].

This SIP to Parlay mapping investigates how in a third generation network a SIP multimedia call control system, called CSCF (Call State Control Function), can be controlled by third party applications running in an un-trusted domain using the Parlay APIs. It is proposed to map the SIP REGISTER message and the Parlay User Status Service in order to make the mobility aspect of using SIP in a wireless network visible to the third party application. Furthermore an extended mapping between SIP messages and the Parlay Generic Call Control Service is given in order to clarify the implementation of the Parlay call control objects.

3.3.1 Parlay Call Processing Services

Parlay offers several kinds of call processing service APIs that differ from each other with respect to the functionality that is offered towards the application program. The Generic Call Control Service provides the basic call control service for the API and allows the set-up, routing and release of calls with two parties. The Multi-party Call Control service enhances the functionality of the Generic Call Control Service with call leg management so that several parties can be attached to one call. The Multi-Media Call Control service enhances the functionality of the Multi-Party Call Control Service with multi-media capabilities that allow control of media channels that are associated with call legs. The Conference Call Control Service finally enhances the Multi-Media Call Control Service by offering routines for the establishment of conference calls.

The Generic Call Control Service (GCCS) is based around a third party model, which allows calls to be instantiated from the network and routed through the network. The GCCS supports enough functionality to allow call routing and call management for today's Intelligent Network (IN) services in the case of a switched telephony network, or equivalent for packet based networks. Although the Parlay GCCS API specification is still somewhat IN oriented it is shown that a complete mapping of GCCS functionality towards Session Initiation Protocol (SIP) [Pailer 01] messages is possible, so that Parlay can be used for control of a modern IP based call processing system.

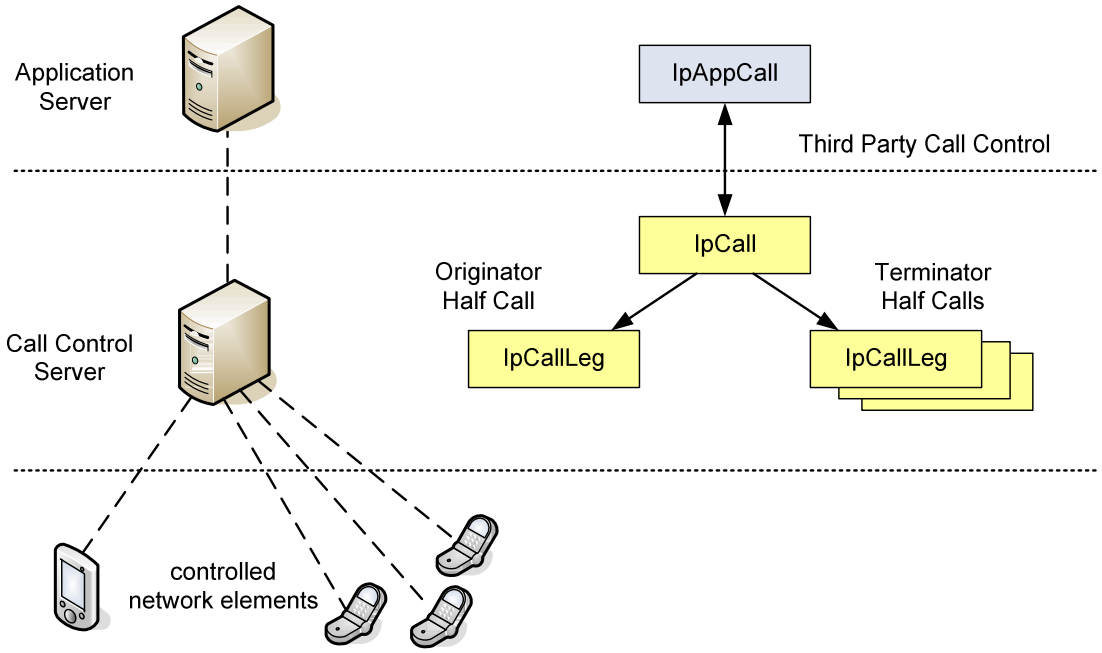


Figure 3-9: Parlay Generic Call Control model

The call model adopted by Parlay defines the following objects (Figure 3-9 and Figure 3-10):

- Call Control Manager (IpCallControlManager): The call control manager controls all active calls in the call processing system. Applications can request to be notified if a call meets certain criteria like call state, originating address or destination address. Furthermore the application can request a new call to be created by the call control manager, which corresponds to a third party call set-up.
- Call object (IpCall): A call represents a relation between a number of parties. The parties in the call are represented by call legs. A GCCS call object can only hold two parties (basic call between an originating and a terminating party). Operations on a call object (e.g., release) will affect all attached call legs.
- Call leg object (IpCallLeg): The call leg object represents a logical association between a call and an address. The relationship includes at least the signalling relation with the party. The relation with the address is only made when the leg is routed. Before that the leg object is IDLE and not yet associated with the address.
- Address (TpAddress): The address logically represents a party in the call.

The Parlay call control objects IpCallControlManager, IpCall and IpCallLeg living in the trusted Network Provider domain are connected to the application in the un-trusted Service provider domain through associated call-back objects (IpAppCallControlManager, IpAppCall and IpAppCallLeg respectively). The application gets a reference to a Call Control Manager object from the Parlay Framework, where authentication and authorisation procedures have to be passed.

There are two ways for an application to get control of a call. The application can request to be notified of calls that meet certain criteria. When a call occurs in the network that meets these criteria, the application is notified and can control the call. Some legs will already be associated with the call in this case. Another way is to create a new call from the application, which corresponds to a third party call set-up.

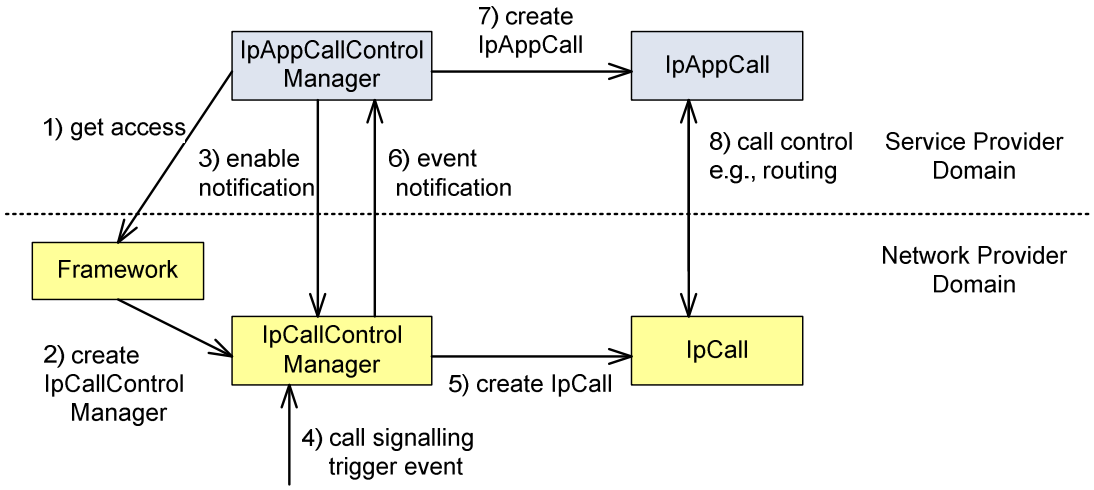


Figure 3-10: Parlay Generic Call Control Message diagram

Notification about calls can be directed to the Application Call Control Manager object or the Application Call object. An event sent to the Application Call Control Manager object results in the instantiation of a Call object. It can be requested by the application that the occurrence of an event will interrupt call processing in the SIP Server.

Parlay notification events will contain the parameters DestinationAddress, OriginatingAddress, Original-DestinationAddress and the RedirectingAddress.

Figure 3-10 shows the message diagram for the handling of notification events. First the Application Call Control Manager (IpAppCallControlManager) contacts the Parlay framework and gets a reference to the Network Call Control manager (IpCallControlManager). The application can now set enable call notifications with trigger criteria like originator address range, terminator address range, monitor mode (interrupting or notifying), type (originating, terminating side), and call event (OFFHOOK, ADDRESS COLLECTED, ADDRESS ANALYSED, ROUTE SELECT FAILURE, BUSY, UNREACHABLE, NO ANSWER, ANSWER). If a call in the network matches the trigger criteria, the Call Control Manager creates a network call object (IpCall) and sends a notification to the Application Call Control Manager, containing the reference to the IpCall object. The application creates the application side call control object (IpAppCall) and has henceforward control over this call.

For the GCCS, only a subset of the general Parlay call control model is used; the API for generic call control does not give explicit access to the legs and the media channels. The GCCS is restricted to two party calls. Access to call legs and multiparty calls is given by the Multi-Party Call Control Service. Explicit control over the media channel is provided by the Multi-media Call Control Service.

The Multi-party Call Control service enhances the functionality of the Generic Call Control Service with leg management. It also allows for multi-party calls to be established, i.e., up to a service specific number of legs can be connected simultaneously to the same call.

Associated with the signalling relationship represented by the call leg, there may also be a bearer connection (e.g., in the traditional voice only networks) or a number (zero or more) media channels (in multi-media networks).

A leg can be attached to the call or detached from the call. When the leg is attached, this means that media or bearer channels related to the legs are connected to the media or bearer channels of the other legs that are attached to the same call. This means, only legs that are attached can 'speak' to each other. A leg can have a number of states, depending on the signal-

ling received from or sent to the party associated with the leg. Usually there is a limit to the number of legs that are in the state of being routed (i.e., the connection is being established) or connected to the call (i.e., the connection is established). Also, there usually is a limit to the number of legs that can be simultaneously attached to the same call.
The Multi-Media Call Control service enhances the functionality of the Multi-Party Call Control Service with multi-media capabilities. To handle the multi-media aspects of a call the concept of media channel is introduced. A media channel is a unidirectional media stream that is associated with a call leg. These channels are usually negotiated between the terminals in the call. The multi-party Call Service gives the application control over the media-channels associated with the legs in a multi-media call in the following way:

- The application can be triggered on the establishment of a media channel that meets the application-defined characteristics.
- The application can monitor the establishment or release of media channels of an ongoing call.
- The application can allow or deny the establishment of media channels (provided the channel establishment was monitored/notified in interrupt mode).
- The application can explicitly close already established media channels.
- The application can request the media channels associated with a specific leg.

The Conference Call Control Service enhances the Multi-Media Call Control Service. The Conference Call Control Service gives the application the ability to manipulate sub-conferences within a conference. A sub-conference defines the grouping of legs within the overall conference call. Only parties in the same sub-conference have a bearer connection (or media channel connection) to each other (e.g., can speak to each other). The application can:

- Create new sub-conferences within the conference, either as an empty sub-conference or by splitting an existing sub-conference in two sub-conferences.
- Move legs between sub-conferences.
- Merge sub-conferences.
- Get a list of all sub-conferences in the call.
- Manipulate the media in a Multiparty Conference Unit (MCU).
- Handle conference policies.

Furthermore, the Conference Call Control Service adds support for the reservation of resources needed for conferencing. The application can:

- Reserve resources for a predefined time period.
- Free reserved resources.
- Search for the availability of conference resources based on a number of criteria.

3.3.2 Parlay Mobility Services

The Parlay User Location service (UL) provides the functionality to allow applications to obtain the geographical location and the status of fixed, mobile and IP based telephony users. The application has the possibility to request periodical user location updates or query the current location of the user.
The Parlay User Location Camel Service (ULC) provides an interface to the existing cellular networks where the position of the user in the mobile network can be requested. An application programmer can request the VLR Number, the location Area Identification and the Cell Global Identification and other mobile-telephony-specific location information.

The Parlay User Location Emergency Service (ULE) is an extension of above location services where the user location can be automatically requested in an emergency.
The Parlay User Status Service (US) provides information about the general status of a user to an application. The application can find out at what address a user is reachable, not reachable or busy, independent of the underlying network technology. The User Status Service is mapped by our group to the SIP REGISTER message so that a Parlay application program can make full usage of SIP mobility features.

3.3.3 Mapping SIP Functionality to Parlay Services

In order to realise the Parlay Service Interfaces, it is recognised that categories of resource interfaces are required to facilitate integration of network equipment. The definition of the resource interfaces is not in the scope of the Parlay forum. The SIP Servlet API will be used as resource interface. This means that the Parlay service interfaces will be implemented as one or more SIP servlets, also called Siplets.
The SIP Servlet API offers HTTP servlet like transaction based access to a SIP protocol stack while hiding mechanisms to keep the protocol reliable (e.g., timeouts and retransmissions) and offering methods for Servlet interaction, Servlet communication and Servlet lifecycle control. Based on the SIP transactions offered by the Servlet Container the Parlay Servlet has to instantiate and control the call control objects such as calls, call legs, and addresses.
Implementing Parlay services as SIP servlets and using Parlay APIs for service control certainly reduces performance by introducing a remote interface between the application and the service logic. It adds however flexibility due to a fully distributed call model, a mature service management framework, compatibility to existing telecom services (PSTN; GSM) and standardized interfaces to other network resources like voice mail systems, location servers, or SMS servers.
Desrochers et al. [Desrochers] have experimented with Parlay over a SIP system by implementing selected services (e.g., wake up call) using the Generic Call Control Service (GCCS) and the Generic User Interaction Service (GUIS) APIs. The result of this work was that Parlay GCCS API specification is still somewhat IN oriented so that not all parameters can be mapped. Furthermore only simple examples were investigated and a complete mapping was not presented.
In principle however it is concluded in [Desrochers] that a mapping regarding the investigated services is possible as long as the controlled SIP server is call-stateful, meaning that call related state information has to be stored during the lifetime of a call. The additional SIP features ‘Caller Preferences’ [RFC 3841] was detected as not accessible via Parlay.
In the following a complete mapping of SIP messages to Parlay GCCS events is shown and it is argued that for a complete mapping of SIP functionality to Parlay the User Status Service (USS) from the Parlay Mobility Interfaces is needed in addition to the GCCS and GUIS APIs and that an extension of the USS API would offer the possibility to implement ‘Caller Preferences’ as well.

3.3.3.1 Mapping of the SIP REGISTER message to the PARLAY User Status Service

In a mobile network user location and user status is a very important piece of information for network management and represents the basis for many powerful location aware applications. The geographical user location is obtained from the mobile network by measuring cell parameters or even by using a Global Positioning System (GPS) receiver in the terminal. The

User Location Service (ULS) Parlay API offers access to the geographical user location data. The mobility package of Parlay offers also the User Status Service (USS) API and it is proposed to map the SIP REGISTER message to the ULS API. The SIP REGISTER message allows a client to let a proxy or redirect server know at which address(es) it can be reached and may also be used to install call handling features at the server (e.g., callee preferences).
The Parlay USS API allows requesting the status of a user identified by an address (SIP URI). There is the possibility to immediately request the user status or to get status information whenever the user status changes. The user status contains information whether the user is reachable, not reachable or busy and what kind of terminal (IP terminal is supported) is in use. The definition of the SIP REGISTER indicates that this message may also contain feature specific information, e.g., special SIP headers describing the users preferences, how an incoming call should be handled. This is functionality unknown to today's telephony system and therefore not included in the Parlay APIs. It is proposed to extend the current definition of the USS API by a generic application information field that can be used to transport additional SIP header information.
Thus support of the Caller Preferences parameters is possible. The callee capabilities are expressed as special parameters in the Contact header field of a SIP REGISTER message. Those parameters define for example preferences like the UA's class (residential, business), the UA's duplex capabilities (full, half, send only), the UA's language, the UA's media capabilities or if the UA is a fixed or mobile terminal.
In order to inform a third party service about the callee capabilities these parameters have to be included in the statusReportRes() method parameters. It is proposed to use the TpCallAppInfo type of the GCCS data type definition that includes a field called CallAppGenericInfo of type String. This string can simply contain the Contact headers from the SIP REGISTER request or a parsed version of the parameter values. The TpCallAppInfo is also part of the notification events sent during an active call, so that the parameters of the caller's preferences expressed in Accept-Contact and Reject-Contact header fields can also be forwarded to the third party application. Thus a third party application logic is able to base a routing decision upon the callee's capabilities and the caller's preferences. Alternatives for expressing caller preferences are discussed in [RFC 3841].

3.3.3.2 Mapping of SIP call control functionality to the PARLAY Generic Call Control Service

The Parlay Generic Call Control Service (GCCS) is used in the FTW SP to control basic SIP call signalling. The GCCS API provides the basic call control service in Parlay. It is based around a third party model, which allows calls to be instantiated from the network and routed through the network. The GCCS also supports standard call control functionality that allows call routing and call management in switched telephony networks as well as in packet based networks. A GCCS call object can only hold two parties (basic call between an originating and a terminating party). Operations on a call object (e.g., release) will affect all attached call legs.
The Parlay call object is used to establish a relation between a number of parties by creating a Parlay call leg for each party within the call. A Parlay call is identified by a Parlay call ID and a Parlay call leg by a Parlay call leg ID. Parlay call IDs and Parlay call leg IDs have to be unique within a Parlay call control manager instance but not necessarily globally unique. Associated with the signalling relationship represented by the call leg, there may also be a bearer connection (e.g., in the traditional voice only networks) or a number (zero or more) media

channels (in multi-media networks). Routing a call leg is done with the Parlay routeReq() method that is mapped to sending a SIP INVITE request to the destination address.
Parlay call legs represent the parties in the Parlay call. A Parlay call leg is an association between an address and a call that may have many call legs connected. Note that a Parlay call leg represents the relationship between an address and a call, whereas a SIP call leg represents a pair wise signalling relationship between two SIP User Agents. A connected Parlay Call between two parties has therefore two call legs, one for the originator and one for the terminator. This Parlay call leg model comes from the PSTN world where a call always involves a central switch dividing the call in call halves. In a connected standard SIP call with two parties the SIP call leg represents the signalling relationship between a local UA and a remote UA. Definition of the local address and remote address depends on whether the UA acts as originator or terminator in the call. The SIP model represents the packet switched network view where media packets are directly exchanged between end-points. Establishing a third party call in SIP however involves a third party control server. The third party control server sets up two signalling relationships, one between the third party control server and the originator, which is first set-up, and one between the third party control server and the terminator [RFC 3725].
SIP is a transaction-based protocol. A SIP transaction consists of all messages from the first request issued by a client towards a server until a final response that is sent back from the server to the client. A SIP transaction is identified by its CSeq number within a single SIP Call Leg. A SIP call is controlled by several SIP transactions that determine the SIP call state. The SIP call leg is identified by the SIP call ID and the local and remote addresses of sender and receiver. Local and remote addresses are found in the SIP message To and From headers. For SIP requests sent from the originator to the terminator and the responses belonging to this request the From header contains the address of the originator and the To header contains the address of the terminator. For SIP requests in the other direction sent from the terminator to the originator and the responses belonging to this request the From header contains the address of the terminator and the To header contains the address of the originator.
A SIP server that implements a Parlay GCCS service has to map SIP messages to Parlay calls and call legs. Therefore the SIP server has to be call-stateful, meaning that the state of the call has to be stored in the server and persists for the duration of the call. A call-stateful SIP server has to make sure that it receives BYE requests ending the call.
Considering the different approaches discussed above, it is clear that no one-to-one mapping between the SIP call model and the Parlay call model is possible. Instead a SIP message has to be analysed according to the SIP call ID, type of the call (standard or third party), the direction of the message (issued from originator or terminator) and type of the message (request or response). Together these parameters allow an unambiguous mapping of a SIP message to a Parlay call leg. The mapping of SIP call IDs to Parlay call IDs can be done by a one-to-one connection for the Parlay GCCS. For multi-party call control systems however mapping is more complex because SIP uses different call IDs for each involved party whereas Parlay uses one call ID for all connected parties. Our group therefore decided to base the software design of the Parlay-SIP mapping class on a call leg class that distributes SIP messages to Parlay call legs.
There are two ways for an application to get control of a call. The application can request from the Parlay call control manager to be notified of calls that meet certain criteria. When a call occurs in the network that meets these criteria, the application is notified and can control the call. Some legs will already be associated with the call in this case. Another way is to cre-

ate a new call from the application, which corresponds to a third party call set-up in SIP according to RFC 3725.
Notification events about calls in the network that meet defined criteria are directed to the Application Call Control Manager. This notification event results in the instantiation of an IpCall object on the server side and an IpAppCall object on the application side. After instantiation of the call objects no further notification events for that criteria must be sent to the application call control manager. For further call control call reports have to be ordered. It can be requested by the application that the occurrence of an event will interrupt call processing in the SIP Server. The SIP Servlet API supports interrupting events.
In Table 3-1 the Parlay notification events that are used to notify the application call control manager (IpAppCallControlManager) object are listed together with the corresponding SIP mapping.

Parlay Notification Event	**SIP message**
OFFHOOK	not applicable to SIP; would mean an empty To: header
ADDRESS_COLLECTED	INVITE received before location service look-up
ADDRESS_-ANALYSED	INVITE request received and location service look-up performed
CALLED_PARTY_BUSY	486 Busy Here,600 Busy Everywhere, 603 Decline
CALLED_PARTY_ UNREACHABLE	480 Temporarily unavailable
NO_ANSWER_FROM_ CALLED_PARTY	180 Ringing is followed by a call termination
ROUTE_SELECT_FAILURE	all 4xx, 5xx and 6xx messages not handled by other events
ANSWER_FROM_CALL_PARTY	200 OK

Table 3-1: Mapping of Parlay notification events to SIP messages

Parlay notification events will contain the parameters DestinationAddress (SIP Request-URI), OriginatingAddress (SIP From header), Original-DestinationAddress (SIP To header) and the RedirectingAddress. SIP does not provide the address where the call was last redirected. This information has to be stored and provided by the SIP Servlet implementing the Parlay APIs. Furthermore the IN specific parameter Call Application Information is given in a notification event. Most of this information is not applicable to SIP. It is however proposed by our group to include specific functionality like the caller preferences in the Call Application Generic Information parameter.
After instantiation of a call object, requests and call reports are exchanged directly between the server side Parlay call object IpCall and the application side IpAppCall object. Events in the network are reported by call reports. The application has to request notification by stating call report criteria. In Table 3-2 the Parlay call reports that are used to notify the application call object IpAppCall are listed together with the corresponding SIP mapping.

Parlay Call Report	SIP message
PROGRESS	100 Trying, 183 Session Progress
ALERTING	180 Ringing
ANSWER	200 OK
BUSY	486 Busy Here, 600 Busy Everywhere, 603 Decline
NO_ANSWER	180 Ringing is followed by a call termination
DISCONNECT	BYE request is received or the SIP server detects the call as released; e.g., by a timeout
REDIRECTED	301 Moved Permanently, 302 Moved Temporarily
SERVICE_CODE	special feature header
ROUTING_FAILURE	All 4xx, 5xx and 6xx messages not handled by other events

Table 3-2: Mapping of Parlay call reports to SIP messages

Parlay supports different address plans to identify parties in a call. It is proposed to map SIP user addressed to the address plan P_ADDRESS_PLAN_URL, although a SIP URI is by definition more general than an URL. As a SIP user ID will however mostly have the format user@host this mapping seems appropriate. Otherwise the Parlay address plans would have to be extended by a new type e.g., P_ADDRESS_PLAN_SIP. Furthermore a Parlay address specifies its presentation rule. In telephony systems it is possible to suppress the presentation of the own directory number at the called users terminal. A similar approach has been specified in [RFC 3323].

For the GCCS, only a subset of the general Parlay call control model is used; the API for generic call control does not give explicit access to the legs and the media channels. Therefore SIP re-INVITE requests are simply ignored in the FTW SP implementation as they do not change the signalling relationship between the end-points but only redefine the media channels with new or changed SDP descriptors. The GCCS is restricted to two party calls, access to call legs and multiparty calls is given by the Multi-Party Call Control Service. Explicit control over the media channel is provided by the Multimedia Call Control Service.

3.3.3.3 Third Party Call Control with SIP

A third party service is understood to be some kind of service logic that is located and runs outside of the trusted domain of the network resources that this service controls.

Third party control interfaces shall have the following important properties:

- The third party control interface offers an abstract view of the underlying network resource so that the features and functionality of the controlled resource can be used without detailed knowledge about the specific realization or implementation.
- The third party control interface offers security management, so that service logic running in an un-trusted domain outside of the network resource can be authenticated and authorized to perform ser-vice specific tasks.

The Parlay API specification fulfils the requirements listed above. Furthermore the Parlay APIs are an industry wide standard, in particular for the control of call processing in Next Generation Telecommunication Networks.

A service platform controls call processing in a SIP domain controller via the Parlay API interfaces in order to realize third party call control services. A call control service running on the service platform will therefore act as a Parlay client, while the SIP domain controller has to implement the Parlay server interfaces. In order to show the interworking between SIP and Parlay the Use Cases 'Routing of a SIP call' and 'Media Streaming' are described in detail.

Use Case 'Routing of a SIP Call'

A routing application gives a user the possibility to configure the routing of incoming calls. The service platform provides a simple interface that offers a method to notify the router application of a new incoming call. A second method is used to inform the router application of the result of a requested routing task. The platform service classes *ServiceRouteNewCall_Manager* and *ServieRouteNewCall_Call* have to implement the Parlay *IpAppCallControlManager* interface and the *IpAppCall* interface respectively. These central classes are responsible for the specialization of the general Parlay functionality towards a well-defined feature, namely the routing of new incoming calls towards a new destination.

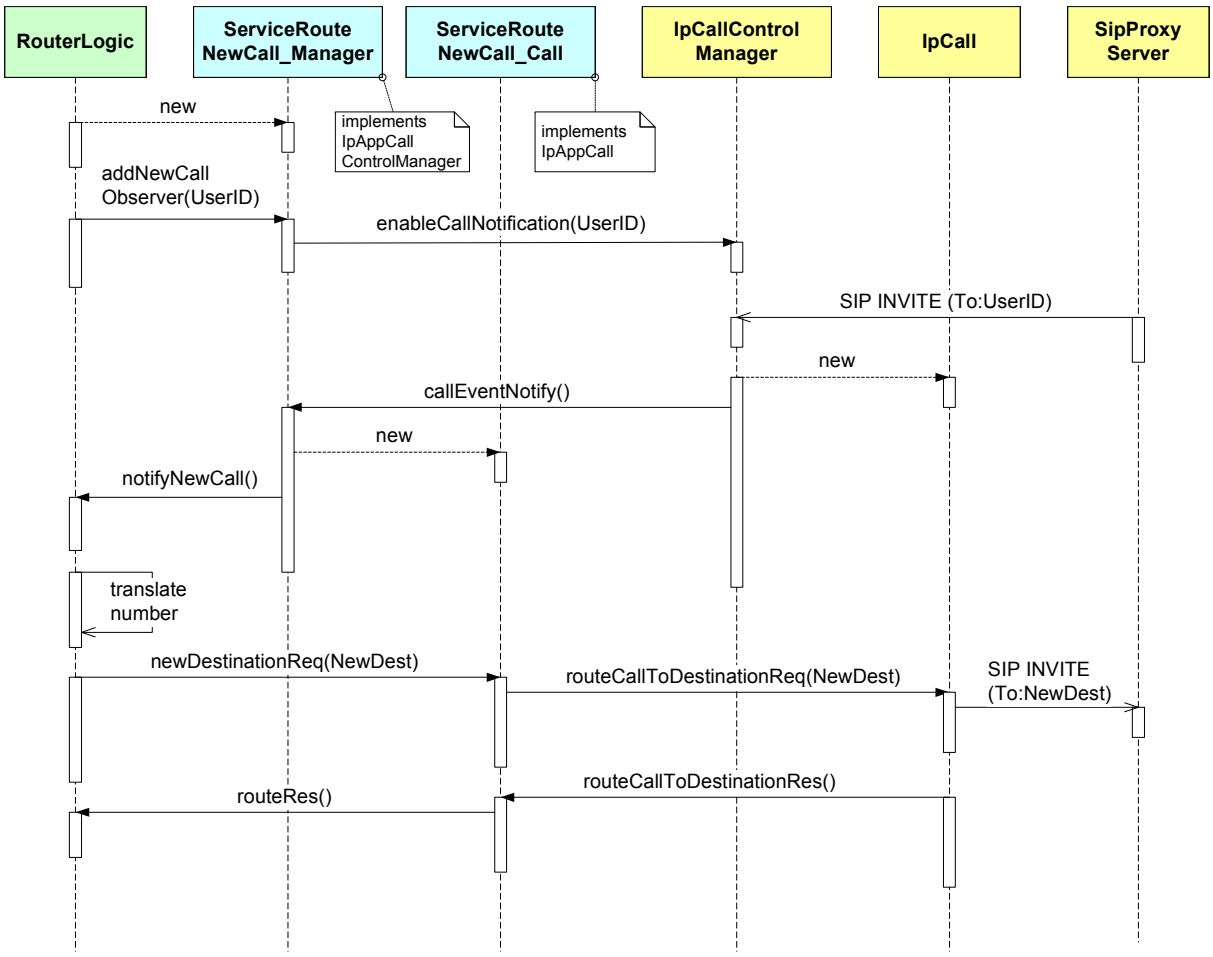


Figure 3-11: Routing of a SIP call

The sequence of events is shown in Figure 3-11 as a message sequence diagram. The following steps are executed:

1. The router application registers to receive notification in case of a new incoming call.

2. The service manager adds the router application to the listed observers and requests the notification from the Parlay server. The Parlay server interfaces are implemented by a SIP proxy server.
3. A new INVITE request is received and the Parlay server notifies the service manager.
4. The service manager creates a service call object and notifies the router logic.
5. The router application computes a new destination for the call.
6. The router application has got together with the notification a reference to the service call interface, where it requests now the new destination for the call.
7. The service call object forwards this request to the SIP server, which proxies the INVITE to the new destination.
8. The Parlay server responds with the result of the routing operation.
9. This result is sent to the routing application.

Use Case 'Media Streaming'

The user is able to receive multimedia streams, such as video-clips or audio-clips, from a content server, located at the third party provider premises and registered at the service platform. The service platform may also offer content adaptation or redirection to pre-rendered content in order to support different terminal types.

Figure 3-12 shows the message sequence diagram of a media streaming session. The user wants to stream down a video clip from a third party content provider. The *iPApp* in the following diagram represents the Parlay client. The user clicks to a link on a website. This causes a HTTP-request. Via a servlet component, the request comes to the Parlay Client, which generates *a routeToOrigination* request. The call setup works in this case like a third party call.

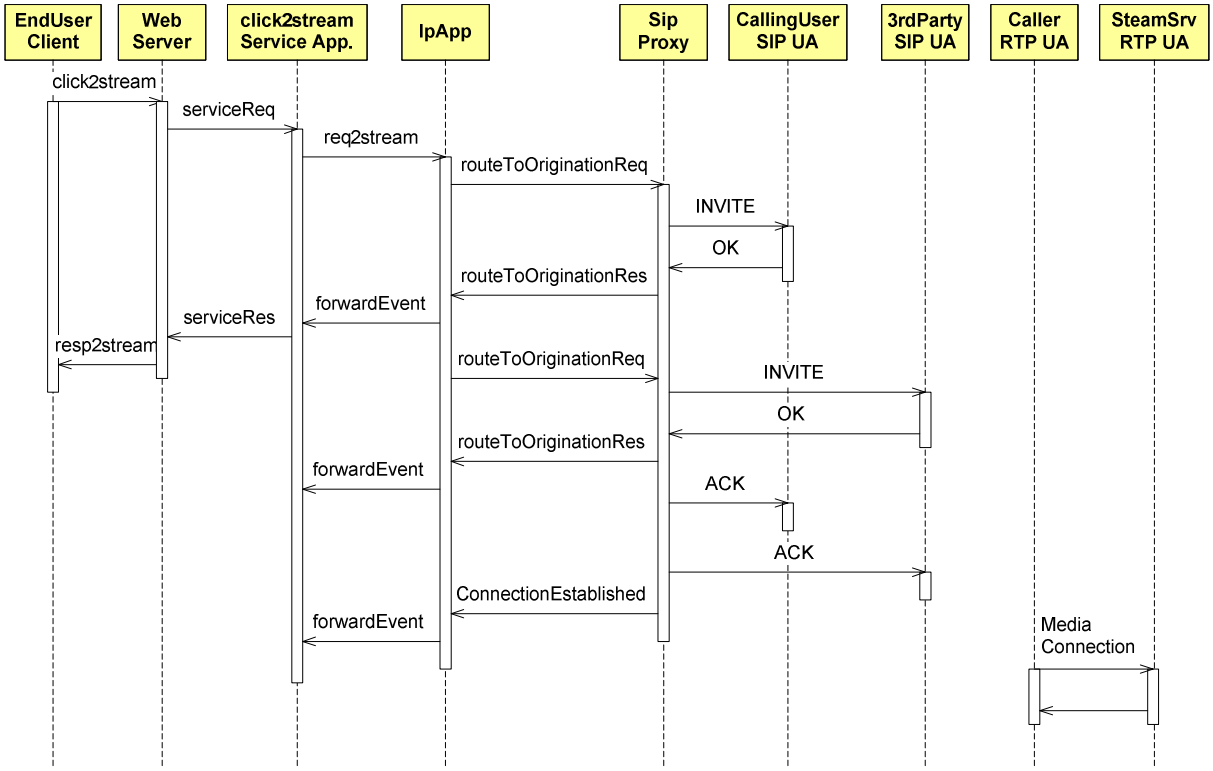


Figure 3-12: Message sequence diagram for the use case 'Media Streaming'

An INVITE request (without IP address and port of the second UA) is sent to the originating UA. After the response (with IP address and port of the originating UA as well as supported media types) the answer to the HTTP request is performed. If the originating UA would not be reachable, an error page could be sent back in the HTTP response. After this a *routeToDestination* request is sent to the SIP proxy, which generates an SIP INVITE with IP and Port of the originator. In the response the destination UA sends back its IP and Port, which reach the originator in the ACK message. After acknowledgment, the service platform is notified about the established connection and the media channel is set up. The transport of the media is established using the Real Time Protocol [RTP].

3.4 Service Oriented Architecture

Web services have introduced the new paradigm of service orientation in the area of distributed computing. This development results in a Service Oriented Architecture [PAP2003], [Erl]. Services are open components that can be easily used by clients to build distributed applications. In order to realize a rapid application development phase, capabilities, interface, behavior and quality of a service are described in a standardized way that is available online. Such service descriptions are published by service-providers towards consumers, that may be business partners or customers that access the service from the Internet, but may also be enterprise internal applications from inside an intranet. An application built from service components can itself act as a service component again, so that modular building blocks can be arranged in different layers of complexity. SOA thus describes a distributed computing architecture that provides both intra- and cross-enterprise application integration and collaboration. The SOA concept is organized in three layers:

- Basic service functionality and discovery
  This layer describes the service interfaces regarding operations and data types, security parameters, the service protocol bindings and the service endpoint identifiers. Furthermore available services are published in a service registry and thus made discoverable by clients. Service discovery is supported by decorating services with information about the service provider, information about the technical service interface and a categorization of the service in a standard taxonomy.
  Service discovery with a priori knowledge about the semantics of a service using the technical service description is the best developed area of the Service Oriented Architecture and is widely used in the industry. Searching for services via autonomous agents (web robots, web crawlers) by classifying services into a taxonomy, a hierarchical tree structure, is still an area of research and certainly enters the field of artificial interpretation of the semantics of human language by expert systems.
- Service composition
  The service composition layer describes the workflow how services shall be used in order to produce a meaningful outcome. Composition declares the sequence of service invocations, defines transactional properties and decides about reactions on possible results that are part of an overall business process. Business process and workflow management is also referred to as service choreography and service orchestration. In the area of service composition, standardization work is in progress, but commercial systems are not widely deployed.
- Service management
  Service management is a pivotal point in a service provider's platform. The management layer has the task to check authentication and authorization of a service consumer, to monitor service invocations, to enforce Service Level Agreements (SLA), to do billing and rating. Furthermore it has to manage the service lifecycle and to ensure that service operation scales to the demand. Service management is not specified by standardization, but an area of competition between different service platform vendors.

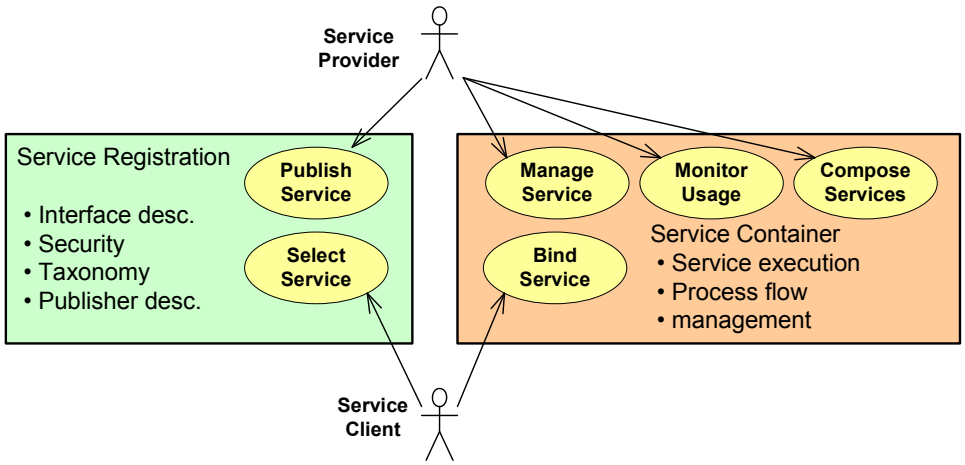


Figure 3-13: Service Oriented Architecture Use Cases

Services in the SOA concept have to comply with the following fundamental principles:

- Services shall be accessible in a platform neutral way.
  Service invocation is independent of the technology used to implement the service. Messages are sent in a platform-neutral, standardized format delivered through the interfaces. XML is the most obvious format that meets this constraint.
- Encapsulation of service implementation details.
  A service is solely described by its interface by machine-processable metadata. No further knowledge about the actual service instance or implementation is necessary for service invocation.
- Message orientation.
  The service is formally defined in terms of the messages exchanged between service and client. The internal structure of service, including features such as its implementation language, process structure and even database structure, are deliberately abstracted away in the SOA. A key benefit of this concerns so-called legacy systems. By avoiding any knowledge of the internal structure of a service, one can incorporate any software component or application that can be "wrapped" in message handling code that allows it to adhere to the formal service definition.
- Service invocation is independent of service location.
  Service locations are discovered in a service registration prior to service invocation.

In general SOA and Web services are most appropriate for applications:

- That must operate over the Internet where reliability and speed cannot be guaranteed;
- Where there is no ability to manage deployment so that all requesters and providers are upgraded at once;
- Where components of the distributed system run on different platforms and vendor products;
- Where an existing application needs to be exposed for use over a network, and can be wrapped as a Web service.

The next sections will discuss an actual realization of the SOA principles by protocols like SOAP, WSDL or UDDI that have been proposed by standard bodies and industry and that are commonly associated with Web services.

3.4.1 Web Services

Like many other technologies of the Internet, Web services have been welcomed by the technical community with a veritable hype. Since then, much of the first enthusiasm has been replaced by insight, how this technology can help to merge the concepts of distributed computing with the power of the World Wide Web.
The term Web services already explains that a web of services is built from distributed service components. These Web services are then arranged into a set of distributed applications and e-business processes.
Web services introduce a new paradigm of how applications interact with a network of services, in the same way as the World Wide Web has revolutionized the way in which people interact with a network of computers.
Web services make use of the highly successful protocols and technologies of the Internet and the World Wide Web (WWW) in order to realize machine-to-machine communications. The next sections will describe the currently used suite of Web service protocols and will discuss their properties, especially regarding the use in service platforms for telecommunications.

3.4.2 Web Service Protocols

Web pages can be created, published and viewed on practically every available platform. This universality comes from the use of HTML/HTTP and was originally targeted at human publishers and human consumers. Since its original inception by the designers in CERN, Web technology has developed tremendously and Web pages are now created not only manually by humans, but instead are generated by CGI scripts, JSP, Servlets or similar technologies. Web robots, Web crawlers and other autonomous software agents search for information in the Web in order to produce digests and indexes. Nevertheless the data is still produced for human consumption.
Web services try to replicate the success of the WWW by building on the same principles and technologies. The goal of Web Service is to create, publish and execute services on practically every available platform in an interoperable way.
Web services realize service oriented computing (including several aspects of SOA) on top of the highly successful Internet protocol suite and Web technologies. The main fundamentals that Web services are built on are the Hyper Text Transfer Protocol HTTP [HTTP 1.1] and the Extensible Markup Language XML [XML 1.0]:

- HTTP is an application-level protocol for distributed, collaborative, hypermedia information systems. It is a generic, stateless, protocol which can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through extension of its request methods, error codes and headers. A feature of HTTP is the typing and negotiation of data representation, allowing systems to be built independently of the data being transferred.
  Web services use HTTP as the main transport protocol.
- XML is a simple, very flexible text format derived from the Standard Generalized Markup Language (SGML, ISO 8879). The goal of XML is to enable generic SGML to be served, received, and processed on the Web in the way that is now possible with HTML. Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere.

One of the main advantages of XML is that it is both human and machine readable. The hierarchical structure enables an automated processing of XML documents (the translation into a machine readable representation is called parsing), whereas the text format is easily understood by humans. Web services make heavy use of XML for data representation and data exchange. Furthermore Web Service XML documents use XML Schemas [XML Schema] to define metadata like structure, content and semantics.

The World Wide Web Consortium (W3C, www.w3c.org) has standardized and established the main Web Service Protocols and Web Service Architecture [WS Arch] that are currently used in academia and industry. SOAP [SOAP 1.2].is an XML based protocol that is used to transfer data in form of XML documents between client and server. SOAP uses HTTP as transport protocol, although other protocol bindings are possible (e.g., e-mail binding). The interface of a Web Service is described in the Web Service Description Language WSDL [WSDL 2.0]. The WSDL description includes data types, message formats, operation signatures, protocol bindings and service endpoint URLs. The Universal Description, Discovery and Integration (UDDI) specifications [UDDI] by the OASIS consortium [OASIS] define a way to publish and discover information about Web services. The core component of UDDI is the UDDI business registration, an XML file used to describe a business entity and its Web services. Conceptually, the information provided in a UDDI business registration consists of three components:

- “white pages” including address, contact, and known identifiers
- “yellow pages” including industrial categorizations based on standard taxonomies
- “Green pages”, the technical information about services that are exposed by the business. Green pages include references to specifications for Web services, as well as support for pointers to various file and URL based discovery mechanisms if required.

SOAP, WSDL and UDDI build the core specifications around Web Service technology. The Web Service Interoperability Organization (WS-I, www.ws-i.org) was founded to overcome interoperability problems that resulted from ambiguous or missing specifications in the Web Service standard documents.

Several standardization efforts are under way that have the goal to extend Web Service core functionality. Most advanced is the work around business process modeling and execution for Web services also referred to as Web service choreography and orchestration. IBM and Microsoft have established the Business Process Execution Language for Web services) [BPEL4WS] that has recently been adopted by OASIS for further development. It is expected that major vendors will soon bring first versions of their BPEL engines on the market.

An overview on the currently standardized Web Service framework is given in Figure 3-14.

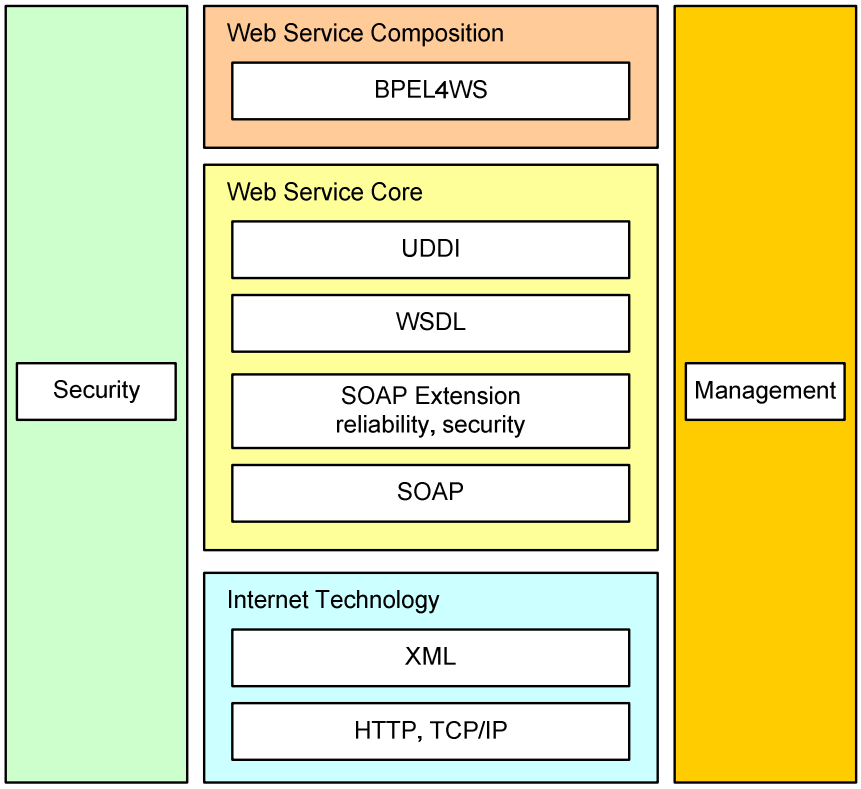


Figure 3-14 Web Service Framework

Additional standardization efforts are undertaken by the OASIS consortium regarding Web Service extensions.

OASIS Web services Security [WS-S] specification describes enhancements to SOAP messaging to provide message integrity and confidentiality. The specified mechanisms can be used to accommodate a wide variety of security models and encryption technologies.

OASIS Web services Distributed Management specification [WSDM] is a committee draft that defines management of any IT resource via Web services protocols (Management Using Web services, or MUWS) and management of the Web services resources via the former (Management Of Web services, or MOWS).

OASIS Web services Reliability (WS-Reliability) [WS-R] is a SOAP-based protocol for exchanging SOAP messages with guaranteed delivery, no duplicates, and guaranteed message ordering. WS-Reliability is defined as SOAP header extensions, and is independent of the underlying protocol.

There are further OASIS technical committees (TC) like the Web services Resource Framework TC, the Web services Notification TC or the Web services Composite Application Framework TC.

The concepts and relationships of W3C's Service Oriented Model (SOM) [WS Arch] are shown in Figure 3-15.

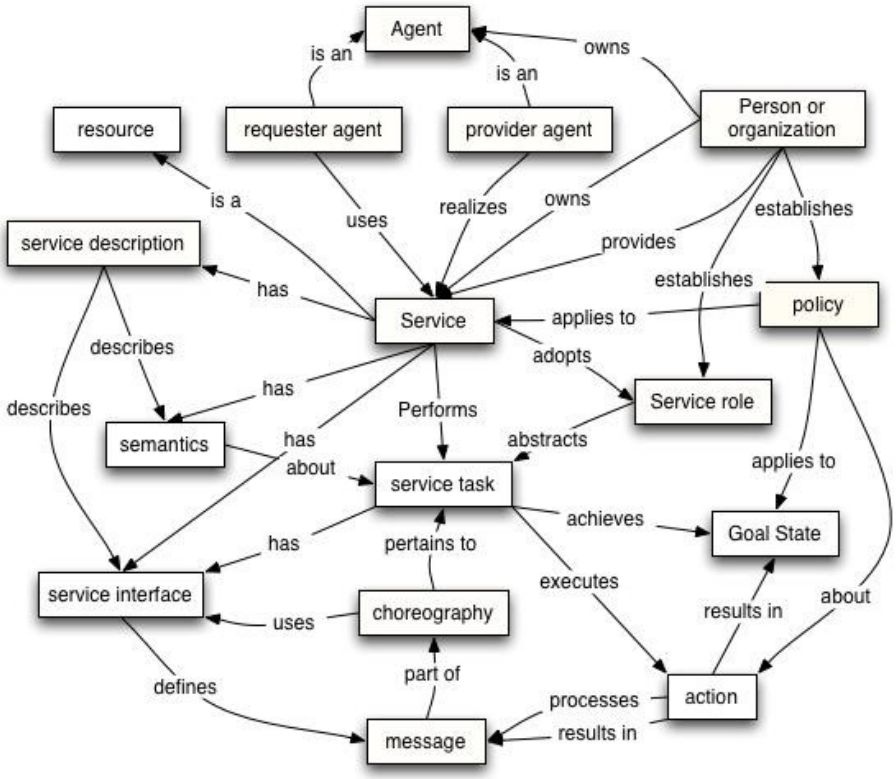


Figure 3-15 W3C's Service Oriented Model

3.4.3 SOAP

The SOAP 1.0 specification was released in 1999 and was largely sponsored by Microsoft. The SOAP 1.1 specification did introduce a framework around the protocol and was submitted to the W3C as a collaborative effort between several companies such as IBM; or Microsoft. Today SOAP Version 1.2 specification is available at the W3C [SOAP1.2].

SOAP provides the definition of the XML-based information that can be used for exchanging structured and typed information between peers in a decentralized, distributed environment. A SOAP message is formally specified as an XML Infoset [XML Infoset], which provides an abstract description of its contents. Infosets can have different on-the-wire representations, one common example of which is as an XML 1.0 document.

SOAP is fundamentally a stateless, one-way message exchange paradigm, but applications can create more complex interaction patterns (e.g., request/response, request/multiple responses) by combining such one-way exchanges with features provided by an underlying protocol and/or application-specific information. SOAP is silent on the semantics of any application-specific data it conveys, as it is on issues such as the routing of SOAP messages, reliable data transfer, or firewall traversal. However, SOAP provides the framework by which application-specific information may be conveyed in an extensible manner. Also, SOAP provides a full description of the required actions taken by a SOAP node on receiving a SOAP message.

The SOAP specification also describes the ways in which SOAP messages may be transported to realize various usage scenarios. Most importantly it describes the SOAP HTTP binding and it introduces two message exchange patterns which are available to an application, one of which uses the HTTP POST method, while the other uses HTTP GET. Examples are also provided on how RPCs may be represented in SOAP message exchanges in a manner that is compatible with the architectural principles of the World Wide Web.

The essence of SOAP is that the protocol is built on a set of open, extensible standards and that simple rules govern the use of those standards. From HTTP and XML, SOAP inherits openness, secure Internet support, robustness and scalability. But also the disadvantages of SOAP find their root in HTTP, XML and XML Schemas. The chattiness or verbosity of SOAP comes mainly from the use of XML and XML Schemas which results in the transport of larger than necessary packages. The higher latency of SOAP requests when compared to other protocols like GIOP in CORBA comes from the statelessness of HTTP and from the necessity to parse XML [Bequet2001].

3.4.3.1 SOAP Message

A SOAP message is fundamentally a one-way transmission between SOAP nodes, from a SOAP sender to a SOAP receiver, but SOAP messages are expected to be combined by applications to implement more complex interaction patterns ranging from request/response to multiple, back-and-forth "conversational" exchanges. Before a SOAP message reaches the ultimate receiver, the message may be routed through several SOAP intermediaries, which act in some way on the message (e.g., security, logging, routing).
Figure 3-16 shows a SOAP message example that contains a request to send a SMS. The message contains the request to send a short message text to a telephone number. A header is used to convey context information containing a UUID for session identification (assuming that the user has authenticated himself and got a UUID during that process).

```
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope
    xmlns:soapenv=”http://schemas.xmlsoap.org/soap/envelope/”
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header>
  <ns1:ContextHeaderType
      soapenv:actor="http://schemas.xmlsoap.org/soap/actor/next"
      soapenv:mustUnderstand="0"
      xmlns:ns1="uri:mobilkom.at/common/types">
   <ns1:uuid >5A389AD2-22DD-11D1-AA77-002035B29092</ns1:uuid>
  </ns1:ContextHeaderType>
 </soapenv:Header>
 <soapenv:Body>
  <SendSmsRequest xmlns=":mobilkom.at/sms/types">
   <PhoneNumber>436641111111</PhoneNumber>
   <SmsText>Hallo Welt!</SmsText>
  </SendSmsRequest>
 </soapenv:Body>
</soapenv:Envelope>
```

Figure 3-16 SOAP message example

A SOAP message is contained in an SOAP Envelope and consists of the mandatory SOAP Body element and the optional SOAP Header element. The content of these elements are application specific. The SOAP Body contains the main end-to-end information. The SOAP header is an extension mechanism that is intended to hold information that is not directly an application payload. SOAP headers typically contain control information for middleware components, SOAP intermediaries, on the way between sender and receiver. Headers may be

inspected, inserted, deleted or forwarded by SOAP nodes encountered along a SOAP message path. Figure 3-17 shows the structure of the SOAP message example.

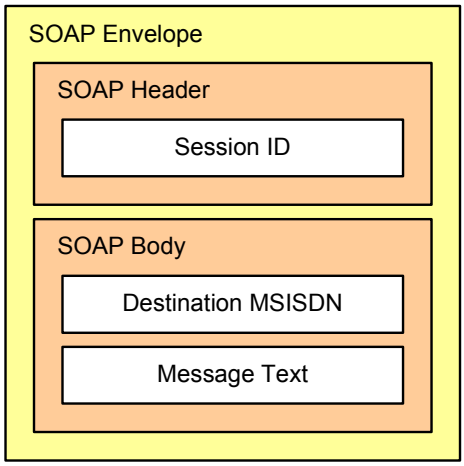


Figure 3-17 SOAP message schematic

The SOAP message example also shows that data types are based on XML Schema definitions [XML Schema]. Complex types can be derived from the built-in datatypes of the XML Schema specification (see Figure 3-18).
SOAP message elements are given qualified names, which means that they are declared unambiguously by using XML Namespaces [XML NS]. Namespaces are identified by a URI and associated with a prefix that binds an element to the namespace. Thus elements of different namespaces can have the same name (but not the same prefix) in one XML document.
It is an important advantage of SOAP, that sent or received messages are XML documents and therefore are human readable. This however comes for the price that the redundancy f the message is increased, and thus is not coded efficiently. Especially XML namespaces can be regarded as verbose.

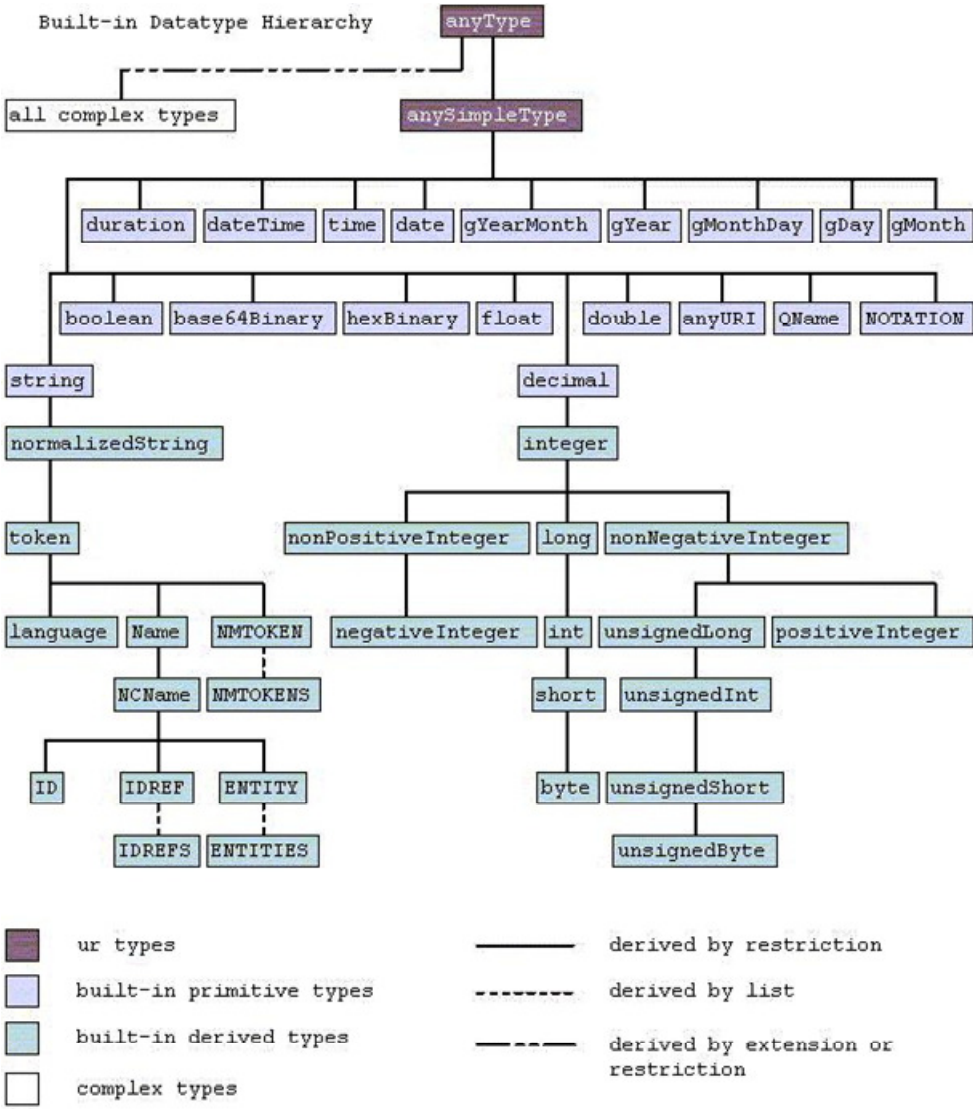


Figure 3-18 XML Schema built-in datatypes

3.4.3.2 SOAP Message Exchange

SOAP is a simple messaging framework for transferring information specified in the form of XML documents between an initial SOAP sender and an ultimate SOAP receiver. The more interesting scenarios typically involve multiple message exchanges between these two nodes. The simplest such exchange is a request-response pattern. It should be emphasized that the use of this pattern as means for conveying remote procedure calls (RPC) does not imply that all SOAP request-response exchanges can or need to be modeled as RPCs. RPC style is used when there is a need to model a certain programmatic behavior, with the exchanged messages conforming to a pre-defined description of the remote call and its return.

SOAP specification uses Message Exchange Patterns (MEP) to describe a complex message flow built from simple SOAP messages. Currently two such MEPs are defined, the *SOAP Request-Response Message Exchange Pattern* on top of which RPCs can be modeled and the *SOAP Response Message Exchange Pattern* which is used to model asynchronous notifications (triggered by non SOAP events). Figure 3-19 shows the state diagram of the Request-Response MEP.

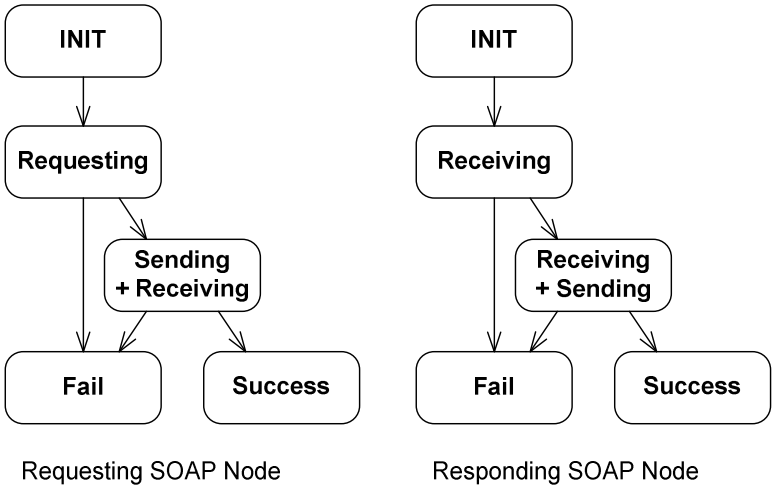


Figure 3-19 SOAP Request-Response MEP State Diagrams

3.4.3.3 SOAP Processing Model

The processing of SOAP messages can be controlled by optional SOAP header elements. A SOAP node (intermediary or ultimate receiver) can assume an application specific role, when processing SOAP headers. The `role` attribute in header indicates that only SOAP nodes in the specified role should act on this header. The default behavior when processing a header is that the header is removed from the message after processing. This default behavior can be overruled by the `relay` attribute. Furthermore the `mustUnderstand` attribute can force the processing of a header which may result in a fault, if the header is not understood.

The SOAP processing model is a very powerful framework for message processing by middleware systems. SOAP headers keep the semantic of service interfaces in the message body simple, while allowing complex processing by intermediary nodes on the way between sender and ultimate receiver. Java SOAP implementations model the SOAP message processing as a handler chain, where a message is dealt with in a sequence of software components that may act on different SOAP headers.

3.4.3.4 SOAP Protocol Bindings

SOAP messages may be exchanged using a variety of underlying protocols, including other application layer protocols. The specification of how SOAP messages may be passed from one SOAP node to another using an underlying protocol is called a SOAP binding. A SOAP message will be made concrete through a protocol binding, whose task, among other things, it is to provide a serialized representation of the infoset that can be conveyed to the next SOAP node in the message path in a manner such that the message can be reconstructed without loss of information. Typically the serialization is that of a well-formed XML 1.0 document. However, there may be other protocol bindings - for example a protocol binding between two SOAP nodes over a limited bandwidth interface - where an alternative, compressed serialization of the same infoset may be chosen.

The SOAP specification defines *features* of a protocol binding that further specify how properties of the underlying protocol are used to support a SOAP message exchange. A typical usage scenario is the request/response message pattern, where a response has to be correlated to request. The widely used HTTP binding supports this feature implicitly, whereas a UDP binding would have to store some kind of transaction ID in a SOAP header.

The SOAP HTTP Binding uses the well-known connection model and a message exchange pattern of HTTP. The client identifies the server via a URI, connects to it using the underlying TCP/IP network, issues a HTTP request message and receives a HTTP response message over the same TCP connection. HTTP implicitly correlates its request message with its response message; therefore, an application using this binding can chose to infer a correlation between a SOAP message sent in the body of a HTTP request message and a SOAP message returned in the HTTP response. Similarly, HTTP identifies the server endpoint via a URI, the Request-URI, which can also serve as the identification of a SOAP node at the server. Figure 3-20 shows the SOAP SMS request example represented by an HTTP message exchange.

```
POST /services/SmsPort HTTP/1.0
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime, multipart/related,
text/*
User-Agent: Axis/1.2beta
Host: localhost:9080
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: "uri:mobilkom.at/sms/service/send_sms"
Content-Length: 759

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope>
 ...
</soapenv:Envelope>



HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Date: Tue, 04 May 2004 13:40:49 GMT
Server: Apache Coyote/1.0

xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <SendSmsResponse xmlns="uri:mobilkom.at/sms/types">
   <ReturnCode>Success</ReturnCode>
  </SendSmsResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

Figure 3-20 HTP representation of a SOAP message

3.4.4 Web Service Description Language – WSDL

The Web services Description Language [WSDL 2.0] is an XML language for describing Web services based on an abstract model of what the service offers. WSDL provides a model and an XML format for describing Web services. WSDL separates the description of the abstract functionality offered by a service from concrete details of a service description such as "how" and "where" that functionality is offered.

WSDL defines the message formats, datatypes, transport protocols, and transport serialization formats that should be used between the client and the server. It also specifies one or more network locations at which a Web service can be invoked, and may provide some information

about the message exchange pattern that is expected. In essence, the service description represents an agreement governing the mechanics of interacting with that service.
WSDL describes a Web Service with a set of *definitions* components. There are WSDL components and type system components. WSDL components are interfaces, bindings and services. Figure 3-21 shows the structure of a WSDL document.

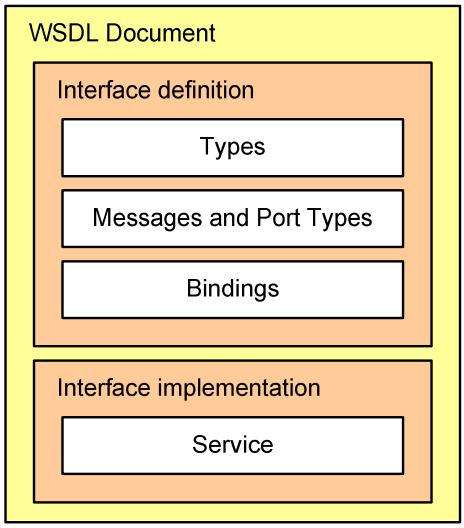


Figure 3-21 WSDL document components

Type system components are element declarations drawn from some type system. Currently the use of XML Schemas is specified as type system and is used to define the local name, namespace name, children and attributes properties of an element information item. Types can be declared directly in the WSDL document or imported from other XML Schema documents.
The WSDL interface component is described by *message* elements and *portType* elements. A request-response exchange is made out of two messages: one for the request and one for the response. A port type then describes the whole operation signature.
The WSDL specification defines bindings, a concrete message format and a protocol for a gives port type. WSDL specifies bindings for SOAP, HTTP and MIME, but is not restricted to those protocols. Other protocols like CORBA, DCOM or RMI can also be supported so that a Web Service could be published that supports more than one protocol by simply adding additional binding elements to the WSDL document.
The bindings for HTTP GET and HTTP POST so that the interaction between a web browser and a web server can be modeled. The “web browser” however can be an application.
The MIME binding is very convenient for providing additional data as part of a message. A weather forecast MMS Web Service might be used to send a satellite photo in form of a JPEG picture to subscribed users. There is the possibility to send the picture data in an XML element (e.g., Base64 encoded). A more elegant method is to use the MIME standard to add the picture data as an additional MIME part to the HTTP message containing the SOAP message.
The *service* element of a WSDL document finally specifies where a service for a specified port type with a given binding can be located. The service location is given as an URL.
Figure 3-22 shows the WSDL description of the SMS sending service that was introduced in the previous section about SOAP. After the declaration of the XML Namespaces, the data types are defined. It is recommended to separate the types definition from the WSDL document, in order to support re-use of data types. In the example the types are imported from the

XML Schema definition file *Sms.xsd* which is shown in Figure 3-23. The element *SmsSendRequest* for example is defined via a complex type with the same name. The SmsSendRequest type is derived from the base type *MessageBase*, which is again imported from another file, and extends the base type by adding elements of *PhoneNumberType* and *SmsTextType* type. The *SmsTextType* is a simple type that restricts the built-in Schema type *string* to 160 characters.

After the type definitions in the WSDL, *message* elements specify the basic building blocks called *parts* of the message exchange. The message parts are used to define operations in the port type section.

The *portType* element defines the service and its operation signatures. It can be seen that the *SmsPort* service supports one operation named *SendSms*, that it accepts one input message named *SendSmsSoapIn* and responses with a *SendSmsSoapOut* message.

In the *binding* section the protocol binding is specified, which is SOAP in our example. The binding may seem repetitive, in comparison to the *portType* section, but defines a concrete realization of the solely logical operational specification of the port type.

Finally the *service* elements specifies the service name and its URL, where a request will be handled.

```
<?xml version="1.0" encoding="utf-8"?>
<definitions name="Sms" targetNamespace="uri:mobilkom.at/sms/service"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:ct="uri:mobilkom.at/common/types"
    xmlns:smstypes="uri:mobilkom.at/sms/types"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:tns="uri:mobilkom.at/sms/service"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <types>
  <xsd:schema elementFormDefault="qualified"
      targetNamespace="uri:mobilkom.at/sms/wsdl"
      xmlns:ct="uri:mobilkom.at/common/types">
   <xsd:import namespace="uri:mobilkom.at/sms/types"
       schemaLocation="sms.xsd"/>
  </xsd:schema>
 </types>
 <message name="SendSmsSoapIn">
  <part element="smstypes:SendSmsRequest" name="aRequest"/>
 </message>
 <message name="SendSmsSoapOut">
  <part element="smstypes:SendSmsResponse" name="SendSmsResult"/>
 </message>
 <message name="SendSmsContextHeader">
  <part element="ct:ContextHeaderElement" name="ContextHeaderType"/>
 </message>
 <portType name="SmsPort">
  <operation name="SendSms">
   <input message="tns:SendSmsSoapIn"/>
   <output message="tns:SendSmsSoapOut"/>
  </operation>
 </portType>
 <binding name="SmsPort" type="tns:SmsPort">
  <soap:binding style="document"
      transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="SendSms">
   <soap:operation soapAction="uri:mobilkom.at/sms/service/send_sms"
       style="document"/>
   <input>
    <soap:body use="literal"/>
     <soap:header message="tns:SendSmsContextHeader"
         part="ContextHeaderType" use="literal"/>
   </input>
   <output>
    <soap:body use="literal"/>
   </output>
  </operation>
 </binding>
 <service name="SmsService">
  <port binding="tns:SmsPort" name="SmsPort">
   <soap:address location="http://localhost:9080/SmsPort"/>
  </port>
 </service>
</definitions>
```

Figure 3-22 WSDL example document

```
<?xml version="1.0" encoding="utf-8"?>
<schema elementFormDefault="qualified"
    targetNamespace="uri:mobilkom.at/sms/types"
    xmlns="http://www.w3.org/2001/XMLSchema"
    xmlns:ct="uri:mobilkom.at/common/types"
    xmlns:tns="uri:mobilkom.at/sms/types"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <import namespace="uri:mobilkom.at/common/types"
     schemaLocation="common.xsd"/>
  <element name="SendSmsRequest" type="tns:SendSmsRequest"/>
  <element name="SendSmsResponse" type="tns:SendSmsResponse"/>
  <complexType name="SendSmsRequest">
   <complexContent mixed="false">
    <extension base="ct:MessageBase">
     <sequence>
      <element maxOccurs="1" minOccurs="1" name="PhoneNumber"
         type="tns:PhoneNumberType"/>
      <element maxOccurs="1" minOccurs="1" name="SmsText"
          type="tns:SmsTextType"/>
     </sequence>
    </extension>
   </complexContent>
  </complexType>
  <complexType name="SendSmsResponse">
   <complexContent mixed="false">
    <extension base="ct:MessageBase">
     <sequence>
      <element maxOccurs="1" minOccurs="1" name="ReturnCode"
         type="tns:ReturnCodeType"/>
     </sequence>
    </extension>
   </complexContent>
  </complexType>
  <simpleType name="PhoneNumberType">
   <restriction base="string"/>
  </simpleType>
  <simpleType name="SmsTextType">
   <restriction base="string">
    <length value="160"/>
   </restriction>
  </simpleType>
  <simpleType name="ReturnCodeType">
   <restriction base="string">
    <enumeration value="Success"/>
    <enumeration value="Failure"/>
   </restriction>
  </simpleType>
</schema>
```

Figure 3-23 XML Schema definition for the SMS request

3.4.5 UDDI

OASIS Universal Description, Discovery, and Integration (UDDI) specification [UDDI API] [UDDI API] is sued to publish Web services. A UDDI registry works similar to a telephone book, where potential business partners or customers can look up companies that are ordered according to the services they offer. In addition UDDI also includes all the information needed to use the service automatically, after it was chosen.

UDDI offer solutions for the following tasks (see Figure 3-24: UDDI use cases):

- Service discovery
  UDDI defines access to a registration repository where companies can register their services and customers can look for services.
- Service description
  XML documents are stored in the registration repository that describe the services and their publishers. Web services are described using WSDL.
- Service integration
  UDDI can be used to store information about any type of software component. Web services will be integrated using SOAP.

The phone book analogon is also used in the organization of the registry into three different areas:

- White Pages
  White pages are indexed by the name of companies and contain information such as company name, business area, or contact information.
- Yellow Pages
  Yellow pages sort businesses by categories. The current version supports three taxonomies:
  - North American Industry Classification System (NAICS)
  - United Nations Standard Products and Services Code System (UN/SPCS)
  - Country codes (ISO 3166)
- Green Pages
  The green pages contain the business and technical descriptions of the Web services offered.

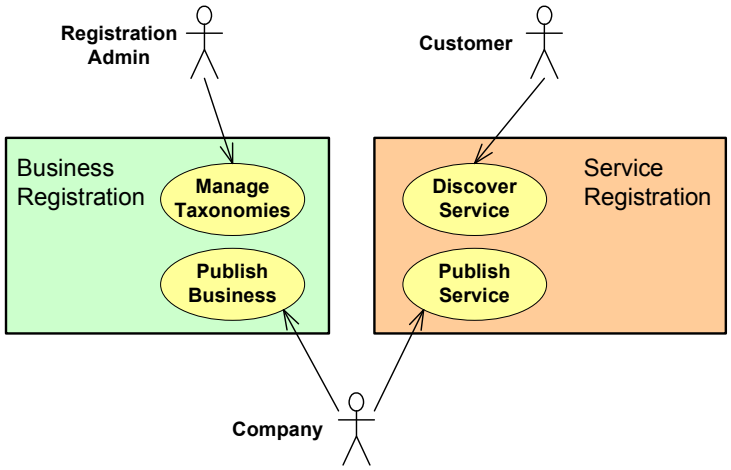


Figure 3-24: UDDI use cases

3.4.6 Business Processes for Web Services

The recently released Business Process Execution Language for Web services (BPEL4WS) specification [BPEL4WS] by OASIS is positioned to become the Web services standard for service composition. BPEL4WS is the successor of Microsoft's "XLANG - Web services for Business Process Design" and IBM's Web Service Flow Language (WSFL).
BPEL4WS defines a notation for specifying business process behavior based on Web services. Processes in BPEL4WS export and import functionality by using Web Service interfaces exclusively.

Business processes can be described in two ways. Executable business processes model actual behavior of a participant in a business interaction. Business protocols, in contrast, use process descriptions that specify the mutually visible message exchange behavior of each of the parties involved in the protocol, without revealing their internal behavior. The process descriptions for business protocols are called abstract processes. BPEL4WS is used to model the behavior of both executable and abstract processes. The basic conceptual model of BPEL4WS links the abstract and executable processes by making it possible to export and import the public aspects embodied in business protocols as process or role templates while maintaining the intent and structure of the protocols. This is arguably the most attractive prospect for the use of BPEL4WS as it greatly increases the level of automation and thereby lowers the cost in establishing cross-enterprise automated business processes.

Web Service *Orchestration* defines the control and data flow between Web services to achieve a business process. Orchestration defines an "executable process (BPEL4WS)" or the rules for a business process flow defined in an XML document, which can be given to a business process engine to "orchestrate" the process from the viewpoint of one participant.

Web Service *Choreography* defines the sequence and dependencies of interactions between multiple roles to implement a business process by composing multiple Web services. Choreography describes the sequence of interactions for web service messages - it defines the conditions under which a particular web service operation can be invoked. WSDL describes the static interface and choreography defines the “dynamic” behavior of the external interface from a global view.

BPEL4WS allows the creation of complex processes by creating and wiring together different activities that can, for example, perform Web services invocations, manipulate data, throw faults, or terminate a process. These activities may be nested within structured activities that define how they may be run, such as in sequence, or in parallel, or depending on certain conditions [Weera 02].

Web services as defined by SOAP, WSDL and UDDI define a loosely coupled interaction and integration model for distributed applications like for example in business-to-business, business-to-customer or enterprise application integration scenarios. Together, these specifications allow applications to find each other and to interact independent off the used platform by following an interaction model that is essentially a stateless model of synchronous or uncorrelated asynchronous interactions.

Models for business interactions typically assume sequences of peer-to-peer message exchanges, both synchronous and asynchronous, within stateful, long-running interactions involving two or more parties. BPEL4WS is used to define such business interactions, a formal description of the message exchange protocols used by business processes. The definition specifies the mutually visible message exchange behavior of each of the parties involved in the protocol, without revealing their internal implementation. Encapsulation of implementation details gives the freedom to change the process implementation without breaking the interaction model and may also be useful to hide internal details of a business process that shall not be disclosed to peers.

BPEL4WS extends the service description model of WSDL by modeling partners and business processes. BPEL4WS process definition provides and/or uses one or more WSDL services, and provides the description of the behavior and interactions of a process instance relative to its partners and resources through Web Service interfaces. That is, BPEL4WS defines the message exchange protocols followed by the business process of a specific role in the interaction.

BPEL4WS introduces the definition of Partner Links <PartnerLinkType> to the definitions part of WSDL. Partner Links define roles for the peers in a business process interaction and link them to port types of a classical WSDL description.
The BPEL4WS process itself declares <partnerLinks>, <variables>, <faultHandlers> and defines activities that can either run in sequence (<sequence>) or parallel (<flow>). BPEL4WS contains all the elements of a Turing-complete programming language, like defining branches in a “case statement” (<switch>), defining loops (<while>), executing one of several alternative paths (<pick>), invoking an operation on some Web service (<invoke>), waiting for a message to operation of the service's interface to be invoked by someone externally (<receive>), generating the response of an input/output operation (<reply>), waiting for some time (<wait>), copying data from one place to another (<assign>), defining exception handling (<throw>, <catch>), terminating the entire service instance (<terminate>), or doing nothing (<empty>). All these elements can be combined recursively so that any arbitrarily complex algorithm can be expressed.
Unlike traditional distributed object systems, BPELWS instances are not created via a factory pattern. Instead, instances in BPEL4WS are created implicitly when messages arrive for the service. Instances are identified by some key fields within data messages. For example, if the process represents an order fulfillment system, the invoice number could be the "key" field to identify the specific instance involved with the interaction. Thus, if a matching instance is not available when a message arrives at a "startable" point in the process, a new instance is automatically created and associated with the key data found in the message. Messages can only be accepted at “non-startable” points in a process after a suitable instance has been located. In BPEL4WS, the process of finding a suitable instance or creating one if necessary is called “message correlation”.

3.4.7 Web Services Security

The Web services Security specification by OASIS [WS-S] describes enhancements to SOAP messaging to provide message integrity and confidentiality. The specified mechanisms can be used to accommodate a wide variety of security models and encryption technologies including PKI, Kerberos and SSL. The specification also provides a general-purpose mechanism for associating security tokens with message content, does not mandate a specific type of security token and is designed to be extensible. Web services Security proposes a standard set of SOAP extensions that can be used to implement message content integrity and confidentiality. When securing SOAP messages, various types of threats should be considered. This includes, but is not limited to:

- The message could be modified or read by antagonists or
- An antagonist could send messages to a service that, while well-formed, lack appropriate security claims to warrant processing.

Web services Security use security tokens in combination with digital signatures to protect and authenticate SOAP messages. Digital signatures are used to verify message origin and integrity by demonstrating the message producer’s knowledge of a signature key. Security tokens assert claims, e.g., can be used to assert the binding between authentication secrets and an identity. An X.509 [X509] certificate, claiming the binding between one’s identity and public key, is an example of a signed security token endorsed by the certificate authority.

Protecting the message content from being disclosed (confidentiality) or modified without detection (integrity) are primary security concerns of the Web services Security specification. Message integrity is provided by XML Signature [XMLSIG] in conjunction with security tokens to ensure that modifications to messages are detected. The integrity mechanisms are designed to support multiple signatures, potentially by multiple SOAP actors/roles, and to be extensible to support additional signature formats.
Message confidentiality leverages XML Encryption [XMLENC], in conjunction with security tokens to keep portions of a SOAP message confidential. The encryption mechanisms are designed to support additional encryption processes and operations by multiple SOAP actors/roles.

3.5 Web Services for Telecommunication Service Access

This section will discuss in detail the advantages and disadvantages of using Web Service technology for service access in telecommunication service platforms.
The main result gained from comparing architecture, design and implementation of telecommunication service platforms using standard distributed object technologies and Web services was, that the paradigms of service oriented computing matched more naturally to the requirements of carrier-grade telecommunication service platforms than the patterns of distributed object technology.

3.5.1 Telecommunication Service Platform requirements

The design of value added services for telecommunication service platforms is driven by the following requirements:

- Large number of service consumers
  Typically the number of subscribers of telecommunication network providers are in the order of millions. Services have therefore to scale to this order of magnitude.
- Services are expected to have high availability.
  Availability of 99,999% is typical for classical voice based telecommunication services. Reliability of value-added services may not be required to fulfill such stringent availability figures, but fierce competition on the market will force service providers to reach at least 99,95%, which roughly translates to 4 hours service downtime per year.
- Heterogeneous networks
  Telecommunication service providers have to maintain large networks that grow organically over time and usually consist of equipment from several vendors employing diverse technologies. Backwards compatibility and the integration of legacy equipment is mandatory to keep existent services running and for protecting past investments.
- Fast service development
  Value-added services are often targeted at specific consumer groups (e.g., teenagers, business travelers, or fans of a soccer club). This means that time and costs for service development are restricted, but also the window of opportunity for successfully and profitably operating a service may only be open for a short period of time. Services have therefore to be rapidly built out of existing components in a generic service framework.
- Manageable service life cycle
  A service life cycle starts with development and testing of a new service. Once the service has “production” quality, it has to be deployed on the servers, where it can be consumed by subscribers. Deployment usually involves several physical machines that are clustered and/or load balanced. Subscription and usage of a service by a customer includes functionality like subscription management, AAA, privacy, confidentiality, logging or customer-care and customer-self-care.
  A real world service will have errors, faults or missing quality that will have to be analyzed, debugged and corrected in a new version. Ensuring end-to-end QoS and analyzing bugs in the end-to-end food chain between client terminal, network connectivity, middleware, back-end systems and application server is paramount for a successful service operation. New versions of services will have to be deployed to correct faults or to add additional features. Versioning of services and backwards compatibility with

deployed clients is therefore an important necessity that should already be considered in the design phase.

The research work of our group was targeted at finding the most suitable architecture and the best technology to fulfill these requirements. In the course of these research projects a prototype telecommunications service platform, further on called FTW-SP, was developed that offers Web Service based access to telecommunication network resources.

The distributed nature of large service provider networks mandates the use of communication protocols and distributed systems technologies for the realization of value-added services. Access to these value-added services in the FTW-SP was first implemented using CORBA Components as a standard distributed object technology and then adapted to Web Service technology and a service oriented architecture.

3.5.2 From Distributed Objects to Service Oriented Computing

A distributed system consists of diverse, discrete software agents that must work together to perform some tasks. Furthermore, the agents in a distributed system do not operate in the same processing environment, so they must communicate by hardware/software protocol stacks over a network. This means that communications with a distributed system are intrinsically less fast and reliable than those using direct code invocation and shared memory. This has important architectural implications because distributed systems require that developers (of infrastructure and applications) consider the unpredictable latency of remote access, and take into account issues of concurrency and the possibility of partial failure [Waldo94].

Distributed object systems are distributed systems in which the semantics of object initialization and method invocation are exposed to remote systems by means of a proprietary or standardized mechanism to broker requests across system boundaries, marshall and unmarshall method argument data. Distributed objects systems typically (albeit not necessarily) are characterized by objects maintaining a fairly complex internal state required to support their methods, a fine grained or "chatty" interaction between an object and a program using it, and a focus on a shared implementation type system and interface hierarchy between the object and the program that uses it [WS Arch]

This main advantage of Web services over remote procedure calls (RPC) oriented distributed object systems is founded on the inherent differences between classical *distributed object* technologies like CORBA, DCOM and RMI and internet style Web services. Web services form *service oriented distributed systems*, by exchanging XML documents asynchronously. In this respect the Web Service protocol SOAP is much nearer to message based systems like e-mail or message oriented middleware like IBM's MQ-Series or JMS than to classical RPC technologies [Vogels 03].

RPC based distributed object systems are working with object references that identify instances of an object and that are passed between processes. Objects are instantiated by a factory upon request, a number of operations are performed on the object instance, and sometime later the instance will be released or garbage collected. A special case is the singleton object, which does not go through the instantiate/release cycle. The state of an object is preserved between method invocations, and thus state-full distributed computing is enabled. The interface to an object (e.g., CORBA's IDL) describes the method signatures that can be used for access to an object, listing input and output parameters and fault conditions.

Web services have none of the characteristics of distributed object systems. There is no notion of an object, object reference, factories or life cycle. There is no notion of an interface with methods, data structure serialization or reference garbage collection. The only technology

Web services have is XML documents and document encapsulation. Web services allow much richer information to flow between peers as the encapsulation of exchanged data in XML documents is less stringent than RPC method signatures. There is a slight analogy between a web service and a (very restrictive) singleton object and implementations of Web services in object oriented languages like Java often use a singleton object to model this behavior.
Web services have no notion of state and thus fall into the category of distributed system techniques that enable stateless computing. In Web services the state of the interaction has to be contained within the semantics of the XML documents that are exchanged.
The SOAP specification did include an encoding that was called *RPC/encoded* and intended to define a generic data encoding scheme for RPC like interactions. In SOAP 1.2 RPC encoding is now optional and the preference is given to *document/literal* encoding, that is also the only encoding supported by the WS-I interoperability specification. This change in the specification reflects the insight, that document oriented computing has architectural advantages over RPC based distributed object technology.
Web services realize the architecture and principles of service oriented computing by offering the following abstractions:

- Service
  A Web Service is a software component that is able to process data encapsulated in XML documents. It is not important what technology is used to implement the service nor what platform the service runs on.
- Document
  All application specific data is exchanged and encapsulated in XML documents that are commonly described by WSDL. The data is encapsulated in an *Envelope* that separates application data from protocol data that may be necessary to send documents over the wire (e.g., the HTTP binding of SOAP).
- Address
  Services are addressed by using URLs. The service URL references a service in combination with a specific protocol binding.

The advantage of using Web services in a service oriented distributed system lies in the fact, that the abstractions Service, Document and Address are sufficient to realize platform and technology independent distributed applications whereas the abstraction of distributed object systems with object references and an object life cycle often hinder the implementation of clear, easy-to-use service architectures.
Our research has shown that applications in telecommunication systems do either not need the “statefulness” of distributed objects, or that state is more naturally encapsulated in the semantics of the exchanged data.
Messaging services for SMS, MMS or e-mail are an important part of telecommunication service platforms. A generic message interface allows either to send or to receive message data from one addresser to one or many addressees. Sending a message is a simple stateless interaction, where the application does not care which concrete instance of a message sending object executes the task. Message receiving means that the application has to register a call back reference, where new messages shall be posted. Again the message delivery process is stateless in its nature and the actual object reference of the receiver is of little interest to the message dispatcher. It can even be argued, that a generic inbox Web Service URL makes more sense, as it is able to receive all kinds of messages (such as SMS, MMS, e-mail, voice-

mail), and then instantiates the appropriate handler object. Generally speaking, message interfaces contain all the necessary state in the semantics of the interface (type of message, message data, addresser, addressee, or priority) and implementation details like implementing object instance references are dispensable for the service function.
Call control services offer access for third party applications to a central resource of telecommunication service platforms. A call is modeled by state machines of call legs that represent the parties in a call (e.g., one originator call leg and several terminator call legs for a conference call). A call leg state machine is described by states such as 'idle', 'alerting', 'connected', 'terminated'. From the states of the call leg model and the events like 'hook off', 'set-up' or 'terminate', it can be seen that call control services are by all means state-full. The state however is either implicitly contained in call control messages (e.g., 'AnswerCall'), or is explicitly stated in the semantics of the exchanged messages, like call session IDs or call leg identifiers. The requesting application does again not care what technology is used to implement a call or what the object instance reference of a call leg implementation is, as long as a call control request is delivered to the correct communication session.
It may even be undesirable that object references are disclosed to third party applications when one looks at carrier grade call control engines. Typical architectures of such switching centers implement call legs for different communication protocols as hardware modules with regional processors. Furthermore such protocol modules are usually deployed in pairs, one active one in standby mode, for failover reasons. Call control applications deal with thin wrapping objects that represent the actual instance on a regional processor. In case of a failover, the wrapping object would have to be notified, if the object reference to the wrapper is disclosed to the third party applications, which would add additional and unnecessary complexity to the call control software. Instead call control machines are protocol driven, meaning that call control protocols contain the call session IDs and call leg identifiers necessary to identify the correct call objects via a call manager that is responsible to find the correct objects for a call session. A failover is reported to the call control manager that instantiates a new wrapper object. The Parlay Call Control Service Web services reflect this architecture by defining a call control manager interface and call/call leg interfaces that use session IDs to identify calls and call legs. Interestingly, the Parlay Call Control Web services also define an object reference header in the SOAP envelope which probably comes from converting an interface design for distributed object technology (CORBA) to WSDL. This unnecessary redundancy may be the cause for difficult to track faults, when session ID and object reference do not identify the same object instance.
Our analysis of service interfaces for telecommunication service platforms has shown that statefullness of services is covered in the semantics of service interfaces. For load balancing and failover management reasons it may even be desirable not to disclose object references to external applications. Managing object references thus becomes rather a burden to the client application than a useful feature and can even lead to ambiguous situations where semantics and object reference are getting inconsistent.
Distributed object technology is very mature and robust, especially if its usage is restricted to those environments which it has been designed for: the corporate intranet with often homogenous platforms and predictable latencies. The strength of Web services is in the internet-style distributed computing, where interoperability and support for heterogeneity in terms of platforms and networks are essential. Web services however will need to incorporate some of the basic distributed systems technologies, such as guaranteed, in-order, exactly-once message

delivery or transaction support, in order to be a full technology substitute for today's distributed object systems.

3.5.3 Scalability of Wide-Area Service Access

Telecommunication service access, especially in the case of mobile networks, have to scale to a large number of subscribers, who want to use services, even while roaming in 'visited' networks of network providers that have a cooperation agreement with the home network provider. Service access has to succeed over diverse access networks and technologies like GPRS, UMTS, WLAN, ADSL or LAN with varying network access parameters like latency, throughput or error rates.

Wide-area service access is not only restricted to end-user terminals, but is also an issue for middleware interactions in case of service roaming. Service roaming means that a roaming user wants to use services from his home network or that a roaming user wants to use services offered by the visited network while being charged on his home bill and in coordination with his home profile and home privacy settings.

It thus follows that service access and service interactions cannot make any assumptions about the quality or technology of the underlying transport network and that telecommunication services and middleware will have to establish communication sessions over wide-area networks.

Large scale deployments of RPC based distributed object systems over wide-area networks did not succeed in the past. The reason is that synchronous interactions over wide-area networks are not scalable and that versioning procedure interface at large scale is extremely difficult.

Web services and service oriented architecture show promising results regarding the problems of large scale deployment, although the potential of Web Service technology should not be overestimated. Web services currently are almost exclusively based on SOAP with HTTP binding. HTTP has shown its ability to be the basis of large scale systems in the tremendous success of the World Wide Web. WWW transactions however are designed for human-to-machine interactions where real-time requirements are basically derived from human communications where delays of several hundred milliseconds are not even noticed and response times of several seconds are acceptable. Web services on the other hand are used to implement machine-to-machine interactions, where acceptable response times tend to be smaller, at least by a factor of one or two orders of magnitude, than in human communications. Furthermore it is debatable, whether HTTP over TCP is the transport protocol of choice for high delay, low throughput networks like for example today's mobile networks. SOAP however does only define the asynchronous exchange of one-way messages and does not make any assumptions about the underlying transport protocol. Systems that are based on asynchronous messaging have been successfully deployed at large-scale and especially telecommunication systems do employ asynchronous protocols extensively.

Regarding versioning of services, Web services have a definite advantage over procedural RPC technologies. The method signatures of distributed objects are fixed at compile time. Once compiled the object will response with an error, if the parameters in an RPC do not comply with the interface definition. Web services on the other hand exchange XML documents that are more flexible regarding its structure. XML allows to define optional elements and it depends on the message receiver how elements, that were not formally specified (e.g., in an XML schema) are dealt with. It is therefore much easier to upgrade Web services to newer versions, while keeping backwards compatibility with already deployed clients. Fur-

thermore addressing via URLs offers the possibility to reflect versioning by changing URLs, e.g., by prefixing the path to a service with a versioning part. This technique shows again, that tight coupling between client applications and object instances is undesirable for large-scale systems.
Another possibility for dealing with different versions of Web services is to add version information as SOAP headers to a SOAP message. Intermediary SOAP nodes or SOAP message handlers in the ultimate receiver node can process the version headers and act accordingly, including a change in the message body syntax or semantics.
SOAP intermediary nodes are a powerful instrument to handle large scale SOAP message traffic. SOAP proxies can implement routing, queuing or scheduling schemes and can thus form a SOAP overlay network in order to control load on the ultimate receivers. The flexibility of such a network of SOAP nodes is another advantage of a document oriented message architecture and eases the design and deployment of scalable service implementations.
Finally it has to be taken into account that services for large-scale systems have to be deployed on several physical machines in parallel in order to distribute load between machines and also to enhance service reliability in case of hardware faults. SOAP and HTTP are both state-less protocols which offers the advantage that load balancing becomes trivial, as every request can potentially be served by another physical machine. Statelessness of the protocol does however not imply that the service semantics have to be state-less. It just is out of the scope of the transport protocol to deal with service state.
Web applications like web portals have solved this problem by adding cookies, a session identifying header, to every HTTP request that belong to one session. Alternatively URL-rewriting techniques can be used where the session information is appended to all URLs in the response content by the Web Server so that clients that do not support cookies can be served as well. Consequently however, all HTTP requests containing a specific cookie or session specific URL part have to be associated with an associated session. This can either be done by storing the session state after each transaction in a repository that is accessible to all service instances (e.g., a database) or that each request containing the same cookie is always served by the same physical machine that holds the related session in memory. Load balancers are protocol routers that are able to distribute incoming request according to session information contained in the request. Typically load balancing is based on cookies, session specific URL patterns or simply on the client IP address.
Although SOAP based Web services define an HTTP binding, there is no specification regarding session handling via cookies. Some Web Service frameworks like Microsoft's .NET or Apache's Axis are indeed able to add session cookies to SOAP/HTTP request, but have to rely on the client application that has to copy the session cookie to all subsequent requests. Relying on session cookies in Web Service implementations therefore rather undermines platform independency. It is advisable to base session handling on information contained in SOAP headers or the SOAP body instead. There is an optional SOAP feature that allows adding an *action* parameter (replacing the SOAPAction HTTP header from SOAP 1.1) to the "application/soap+xml" media type which contains an URI. This action parameter can be used to optimize message dispatching and routing, but a server should not require its presence in order to operate. State of the art load balancers are indeed able to act upon SOAP action parameters or to inspect routed HTTP packets for session identifiers (e.g., a SOAP header).
Sometimes the opinion is put forward that the main reason for tunneling Web Service messages through HTTP is to bypass firewalls. If this would indeed be the reason than it would be a dangerous approach that would seriously weaken a site's security, and one should only do

this in combination with extensive content-based filtering techniques of the HTTP flows. Such statements however reflect the daily experience of developers when confronted with corporate intra- and Internet firewall administration. It is in practice almost impossible to get a static TCP port range for RPC connection outside of a corporate intranet (unless tunneled over high security VPNs), not mentioning dynamic TCP port allocation. Opening HTTP/HTTPS ports on the other hand is daily business that is obviously thought to be well understood by administrators and IT executives. This probably comes from the opinion that threats from an open TCP port 80 can be better defended than attacks on other ports. Time will show if security implications of tunneling Web Service requests in HTTP requests through corporate firewalls will open a new front against hacker attacks or whether Web services Security or other security schemes will be sufficient to protect Web Service infrastructure against malicious usage.

Web services are based on HTTP package processing and thus fit well in the already installed Web infrastructure of network and service providers. Application servers, Web servers, load balancers, firewalls, administration and monitoring equipment and other network resources have been developed to cope with exponentially increasing WWW traffic. It is an economic but also technological advantage of Web services to build on the lessons learnt from scaling Web portals to millions of subscribers. In contrast to the period of the dot.com boom, most enterprise software projects currently require a short-term return on investment. This forces most of the production web service projects to focus on improving the access to the corporate data and services for partners and customers, without requiring too much new infrastructure. The first place this is possible is by using the web servers that are already functioning as front end servers. This approach has become rather successful and should be seen as the first step in the path to a deeper integration of Web services in the enterprise.

3.5.4 Service Lifecycle

For a telecommunication service provider it is of utmost importance to deliver services in time and with good quality to the market. The service lifecycle includes development, testing, deployment and operation.

Development kits for Web services have reached maturity in a very short amount of time. This has certainly been helped by the fact, that major vendors of web servers and application servers have added Web services to their portfolio and development environments. The same applies to the deployment and operational phase of a Web Service, where existing Web infrastructures has been extended to support the new technology.

The efforts that have been made in the direction of Web Service interoperability by the WS-I organization certainly pay off. A WS-I compliant WSDL interface is easily converted to a Web Service server skeleton or client stub, regardless of the target platform and with a high probability of correct inter-working.

The verbosity of text based SOAP messages in comparison to RPC technologies that use binary formats has often been criticized to be the cause of performance loss. It cannot be denied that a SOAP message uses more bandwidth when sent over the wire, although the main performance disadvantage of Web services comes from the parsing of XML documents before they can be processed. When it comes to debugging however, a human readable message format has its definite advantages. One glimpse at a the packet trace of a faulty Web Service transaction often reveals the source of an error, which in most cases lies in the semantics of the message, whereas binary formats require tool support or having an eye for patterns in hexadecimal number prints.

It was already mentioned in the discussion about service scalability issues above, that service oriented computing offers more possibility in dealing with versioning of services and backwards compatibility with already deployed clients. The separation of actual service implementation object instance and the service endpoint URL identifier on one side and the flexibility of XML message content on the other hand are key advantages when dealing with service upgrades and fault corrections.

3.5.5 Web Service Components

Time and costs spent for the development of new services in telecommunication service platforms are crucial factors for the success of service providers. Therefore the assessment of service architecture and service implementation technology has to consider how the service development process is influenced by a technological choice. Service design is certainly sped up by the re-use of already existing service components. In the following it shall be discussed how Web services employ the concept of component re-use on several levels of the service design process.

A service endpoint of a Web services is identified by a service URL and this is the most obvious component in a Web Service oriented design. Composition of Web services out of other Web services is for example described by business process languages like BPEL4SW. The service design has therefore to consider the re-usability of service endpoints. The Parlay APIs [Parlay] model of a telecommunication network sees the different network resources like the switch for call control, SMS/MMS gateways for messaging, the GMLC for location data, or the HLR for user status information as service components.

Apart from the structure and granularity of the service endpoint design, there is also a service endpoint implementation. Target languages and target systems vary and it is the strength of Web services to define a platform and technology independent abstraction of concrete service implementations. There is however a component oriented design technique that shall be called *Message Handler Chain* that seems to be emerging in many of the most popular Web Service development platforms. A message handler is an object that is invoked by an incoming or outgoing SOAP message and performs some actions like changing the content of the message, adding some header, or simply writing data to a log file. Message handlers can be arranged in chains where the service endpoint is a special handler at the end of the input chain and at the same time at the beginning of the output chain. Figure 3-25 shows the input and output message handler chains in the engine of the Axis [Axis] framework of the Apache Software Foundation [Apache]. A transport handler is another example of a special handler that abstracts from the used transport binding (e.g., HTTP). The request and response handlers in the service part are implemented by the service designer and are specified by a deployment descriptor.

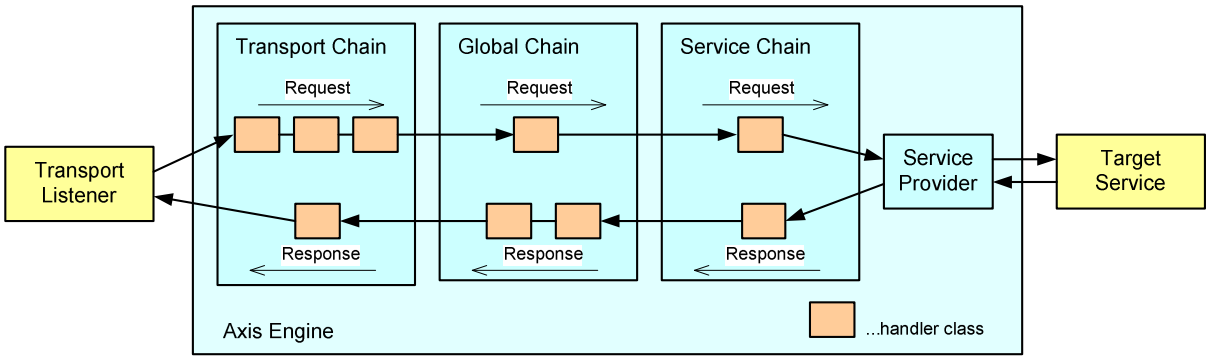


Figure 3-25 Axis Framework Message Path

Sun Microsystems has specified in the Java Community Process JSR-101 “Java API for XML-Based RPC (JAX-RPC)” [JAXRPC] a similar design and thus standardized the Message Handler Chain pattern for the Java programming language. The Axis project is indeed working on support for JAX-RPC handlers. Figure 3-26 shows the class diagram of the message handler design in the JAX-RPC specification.

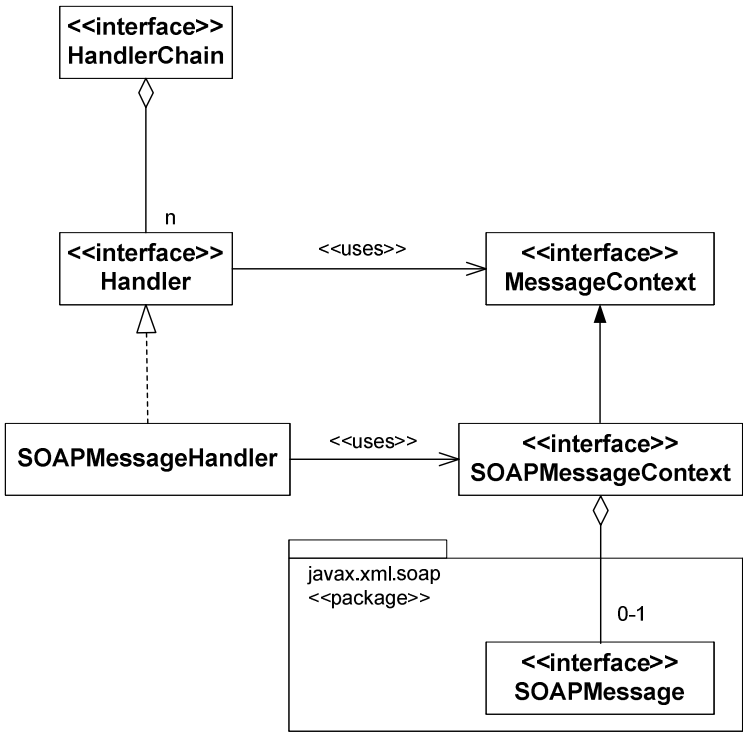


Figure 3-26 JAX-RPC class diagram of a Message Handler Chain

Also Microsoft’s .NET platform offers the possibility to inspect and modify SOAP messages during processing by implement the abstract `System.Web.Services. Protocols.SoapExtension` class [Gitman]. To build a message handler chain on top of this entry point is however left to the designer.

The Message Handler Chain pattern is a variation of the *Chain of Responsibility* pattern [Patterns]. This pattern avoids coupling the sender of a request to its receiver by giving more than one object a chance to handle the request. It chains the receiving objects and passes the request along the chain until an object handles it. The pattern frees an object from knowing which other object handles a request. An object only has to know that a request will be handled “appropriately”. Both the receiver and the sender have no explicit knowledge of each other, and an object in the chain does not have to know about the chain's structure.

As a result, Chain of Responsibility can simplify object interconnections. Instead of objects maintaining references to all candidate receivers, they keep a single reference to their successor.

A Chain of Responsibility adds flexibility in distributing responsibilities among objects. Chains can be dynamically rearranged during run-time or handlers can be statically specialized by inheritance.

The Message Handler Chain construct allows the Web Service designer to implement reusable handler components for tasks like Web services security, monitoring, logging, alerting

or AAA. Message handlers are a generic approach to implement Web Service management in general.
Another important component technology used in SOAP based Web services is the distribution of SOAP message processing between several nodes in the network. The processing of SOAP messages can be controlled by optional SOAP header elements. A SOAP node (intermediary or ultimate receiver) can assume an application specific role, when processing SOAP headers. The SOAP processing model is a very powerful framework for message processing by middleware systems. SOAP headers keep the semantic of service interfaces in the service endpoint simple, while allowing complex processing by intermediary nodes on the way between sender and ultimate receiver. Thus SOAP proxies can act as a placeholder for the ultimate receiver while spanning a SOAP overlay network which implements for example message routing and queuing or security tasks like encryption at the edge of trusted networks.

3.5.6 Parlay Adapter Design of the FTW Service Platform

It was the goal of the FTW SP project to develop a telecommunication service platform that offers access to services via the Parlay APIs [Parlay]. In order to compare distributed system technologies the design should support the CORBA Components Model [CCM] and Web services for third party application access. The CCM system was developed by our group during FTW project A1 [FTW A1] based on OpenORB [OpenORB], a Java based CORBA implementation. The Web Service interface was implemented in the Axis [Axis] framework of the Apache Software Foundation [Apache].

3.5.6.1 Component Containers for Parlay Services

A Parlay service is composed of a service manager that acts as a factory for service instances. The Parlay Framework service is the central service that is used as a factory for service manager instances of other services. For the CORBA version of the FTW-SP, the Parlay Framework service and the Parlay Service Managers are implemented as CORBA components according to the CORBA Component Model. The lifecycle and access of a CORBA component is controlled by the CCM container. A Parlay service implements a skeleton object, an object that is remotely accessible via CORBA, and uses stubs, local representations of remote objects to invoke methods. CORBA uses the Internet Inter-ORB Protocol (IIOP) over TCP/IP to transmit RPCs over the wire (Figure 3-27). It shall be emphasized that a Parlay application uses remote service interfaces as a client, but has also to implement interfaces for callbacks by the service, so that every Parlay Application does not only have to use service stubs but also has to implement callback interface skeletons.
The Web Service implementation of the FTW-SP uses SOAP service implementation objects that are created out of the WSDL service description by the Axis framework. The Axis framework itself runs as an HTTP servlet in any Web or application server that implements the HTTP Servlet API specification [Servlet]. The Web server receives HTTP requests and forwards those with URLs matching to the Axis Servlet and to the Axis SOAP engine, where the SOAP envelopes of the packet are parsed. A service implementation is identified by the Web Service name and called by the Axis engine. During deployment of the service in the Axis framework, the type of object instantiation for the service implementation can be specified in a deployment descriptor XML document.

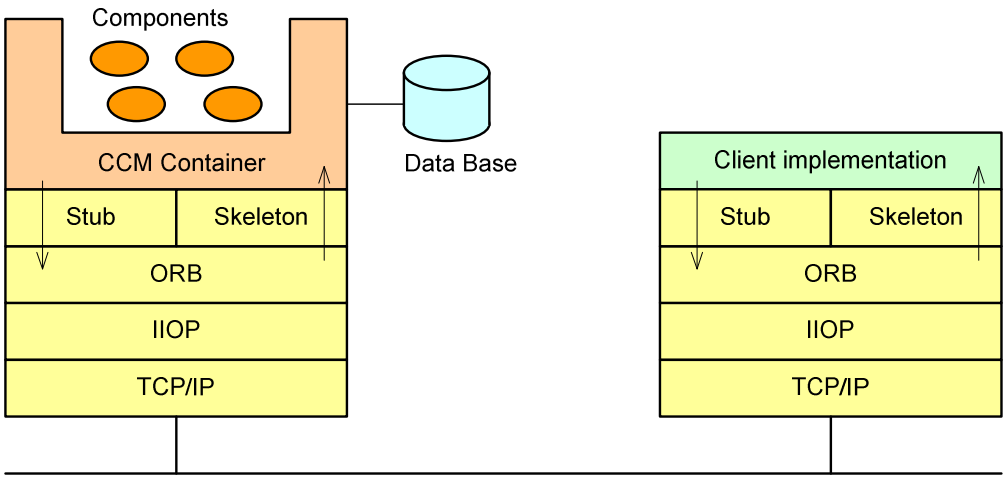


Figure 3-27 CORBA Component Model Container

Axis supports instantiation of the service implementation Java objects in a “Request”, “Session” or “Application” scope. "Request" scope will create a new object each time a SOAP request comes in for the service. "Application" scope will create a singleton shared object to service all requests. "Session" scope will create a new object for each session-enabled client who accesses your service. The Parlay services in FTW-SP were deployed in “Application” scope, because session information is transported in the semantics of the Parlay interfaces (e.g., as a session ID) and thus the service singleton object acts as a message dispatcher for incoming Parlay requests. Figure 3-28 shows the SOAP service implementation objects inside of the Axis SOAP engine that uses the HTTP implementation of an HTTP servlet engine. It shall be noted that the design choice of the Axis framework to implement the SOAP engine as an HTTP servlet is not mandated by the Web Service specifications. It is however a considerable advantage of a servlet based SOAP design, that it can be deployed on already existing Web infrastructure and runs seamlessly on any vendor’s Web- or application server.

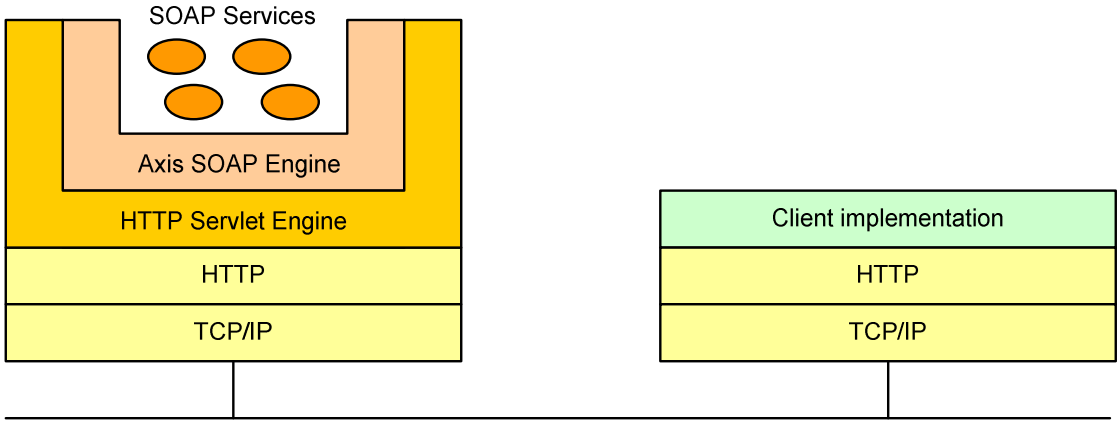


Figure 3-28 AXIS Servlet Web Service Engine

3.5.6.2 The Parlay Adapter - Integrating SIP Call Control with a Component Container

Call control interfaces are an important part of the Parlay APIs and call control is also an integral part of telecommunication service platforms. The FTW-SP uses SIP as its call control protocol (see section *Session Initiation Protocol (SIP)*) and a SIP call control server was developed as part of the FTW A1 project. The SIP call control server implements the SIP Serv-

let API [SIP Servlet], where call states are controlled by pluggable SIP servlets, also called siplets. The SIP Servlet API bears close analogy to the HTTP Servlet API, although differences come from the fact that call control is an interaction where both peers have to implement a client and a server side, whereas Web interactions are strictly client-server based.
It was a main research goal of the FTW A1 project to map SIP call control to the Parlay Call Control Service (see section *Mapping SIP to Parlay*). A reference implementation of the proposed mapping between SIP and Parlay call control was added to the FTW-SP. The SIP implementation of the mapping was realized in form of a Parlay siplet, a SIP servlet component that allows third party applications to remotely control calls in the FTW SIP server.
In order to compare different distributed systems technologies, it was necessary to design the call control service independently of the technology used by an application for remote access. Thus the Parlay call control siplet should either be accessible via CORBA components or via SOAP Web services. Abstraction of the remote access transport technology was done by introducing a set of classes that wrapped the Parlay interfaces, called Parlay Adapter. The Parlay Adapter connects the Parlay siplet either to a CCM container (see Figure 3-29) or to a SOAP engine (see Figure 3-30).

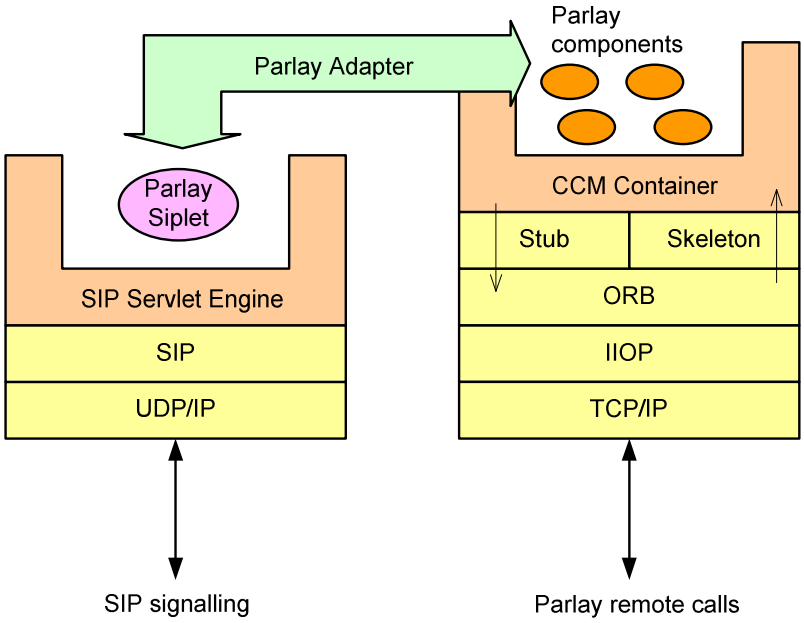


Figure 3-29 Parlay Siplet connection to a CCM Container

It was decided that SIP implementation and remote access implementation (either CORBA or SOAP) should run in the same JVM. Integrating SIP Servlets with either AXIS SOAP services or with CORBA components however is not trivial. It has to be considered that each engine or container has its own bootstrapping mechanism, parameter handling strategy and object lifecycle management. The Parlay Adapter has therefore to make sure that both sides, SIP call control and Parlay remote access, are correctly initialized and can be connected to each other.
The `ParlaySiplet` class was consequently made abstract and is extended by two inheriting classes `CCMParlaySiplet` and `WebServParlaySiplet` considering and encapsulating the special requirements of the concrete remote access technology. The configuration file of the SIP servlet engine states the concrete Parlay siplet class that shall be loaded on start-up, but uses only references to the abstract `ParlaySiplet` class afterwards. The `ParlaySiplet` uses an abstract factory, the `ParlayAdapterFactory`, to instantiate

objects implementing the Parlay service interfaces. The concrete implementation of the factory depends again on the remote access technology, as well as and the objects that this factory instantiates. Access technology details however should be invisible to the call control logic.

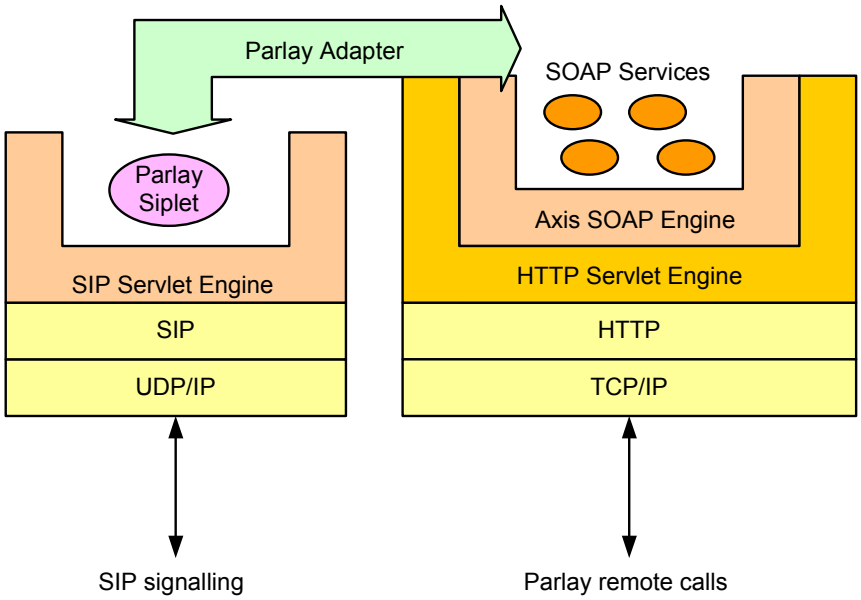


Figure 3-30 Parlay Siplet connection to an Axis Servlet Web Service Container

In order to realize independence of access technology, the Parlay Adapter uses, apart from the *Adapter* pattern [Patterns] that it is named after, several other patterns of object oriented design.

The Parlay service interfaces of the Parlay Adapter (e.g., the `IpCall` or the `IpCallControlManager` classes) implement the Adapter pattern, also known as Wrapper, in the sense that they separate the Parlay interfaces that the `ParlaySiplet` works on from the Parlay interfaces that are generated out of the Parlay interface specifications by tools for different access technologies. The Parlay consortium provides interface specifications for CORBA in IDL and for Web services in WSDL (supporting RPC encoding and document/literal encoding). The *Idl2Java* compiler of the OpenORB package and the *WSDL2Java* compiler of the Axis framework have been used to generate Java skeletons and stubs for the respective technology. The IDL to Java mapping is specified by the OMG in the "IDL to Java Language Mapping Specification" [IDL2Java]. The WSDL to Java mapping has only recently be specified by Sun Microsystems in the Java Community Process JSR-101 "Java API for XML-Based RPC (JAX-RPC)" [JAXRPC]. Both mappings may be very similar, at least from a pure interface perspective, but underlying implementation and helper classes are certainly different. The wrapping of Parlay service interfaces in order to separate call control logic from a concrete remote access technology has additionally the advantage, that different versions of interfaces can easily be supported. The Parlay interface wrappers thus also separate version updates of the access from version updates of the call control logic. Figure 3-31 shows an example of the Adapter pattern for the Web Service implementation class of the Parlay `IpCall` interface. The `IpCallBindingImpl` class implements the `IpCall` interface that was automatically generated by the WSDL2Java compiler. The Web Service implementation does not directly access the `PCallFacet` that implements the call control logic, but instead uses an Adapter, the `IpCall` interface from the `at.ftw.sip.ParlayAdapter` pack-

age. The Adapter interface is implemented by the `WSSklIpCall` class that delegates call control requests to the `PCallFacet` class.

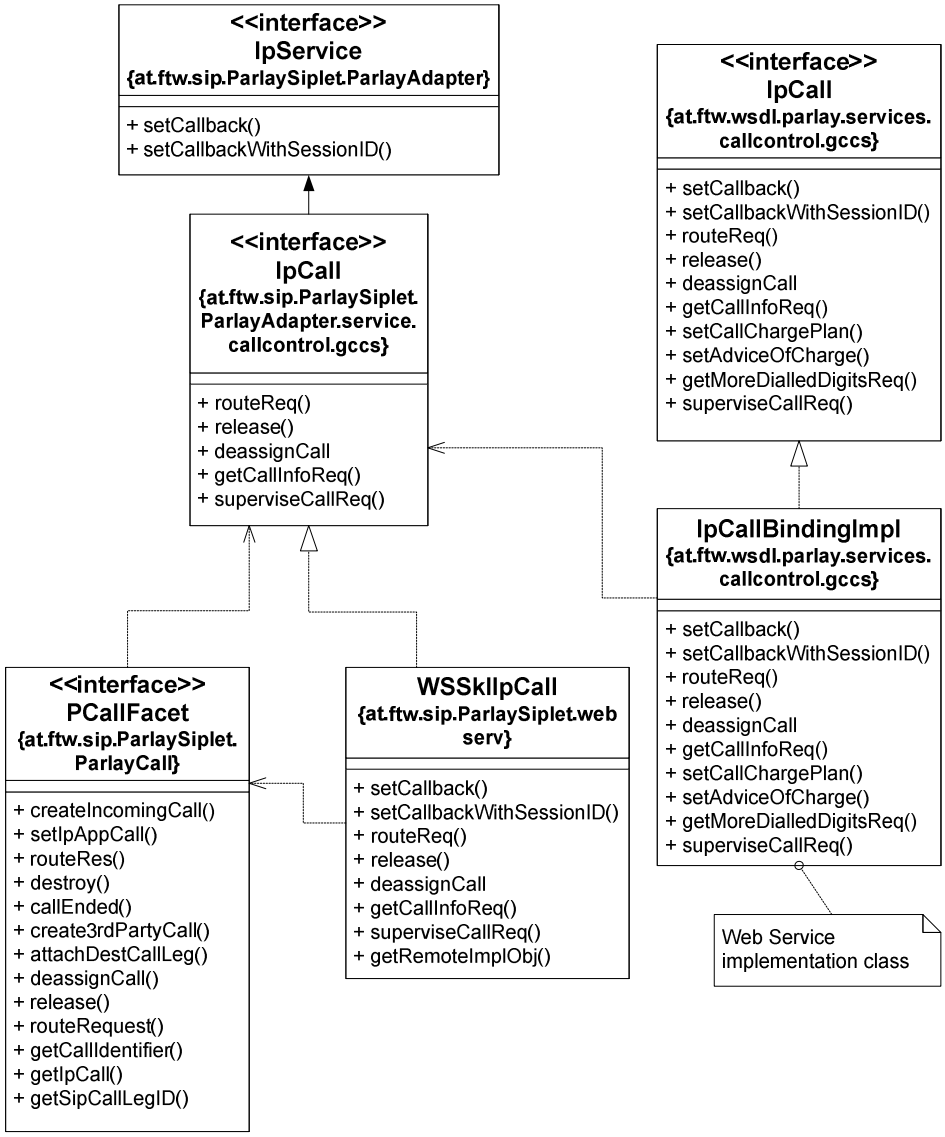


Figure 3-31 Adapter pattern in the Parlay Adapter design

In Figure 3-33 the design of the Parlay Adapter is shown in an overview on the main classes of the Parlay call control service. Depending on the remote access technology either a `WSParlaySiplet` or a `CCMParlaySiplet` class is instantiated by the SIP Servlet engine. During start-up the Parlay siplet is connected to the CCM container via the `CCMSipTransactionComponentImpl` class for CORBA or via the `PSipletAxisServlet` for SOAP. The *Abstract Factory* pattern [Patterns] is used in the design of the `ParlayAdapterFactory` to create Parlay Adapter objects independently of the used remote access technology. An Abstract Factory only declares an interface for creating objects. It is up to the concrete factory classes, `CCMParlayAdapterFactory` and `WSParlayAdapterFactory`, to actually create specific Parlay service objects. The concrete factories also implement the *Singleton* pattern, ensuring that there is only on instance that provides a global point of access. The Parlay service objects that are created by the abstract factory are themselves again abstract interfaces or adapters to the concrete technology dependent

implementation. Figure 3-33 shows as an example for the created classes the `IpCallControlManager` and the `IpCall` interfaces. Parlay connects each service implementing class with a corresponding stub representing the remote call-back class on the application side. The `IpAppCallControlManager` and the `IpAppCall` in the diagram are the local stubs of the application call control objects. Each of these abstract call control classes has concrete implementations for the CORBA Component Model and for the Web Service model.

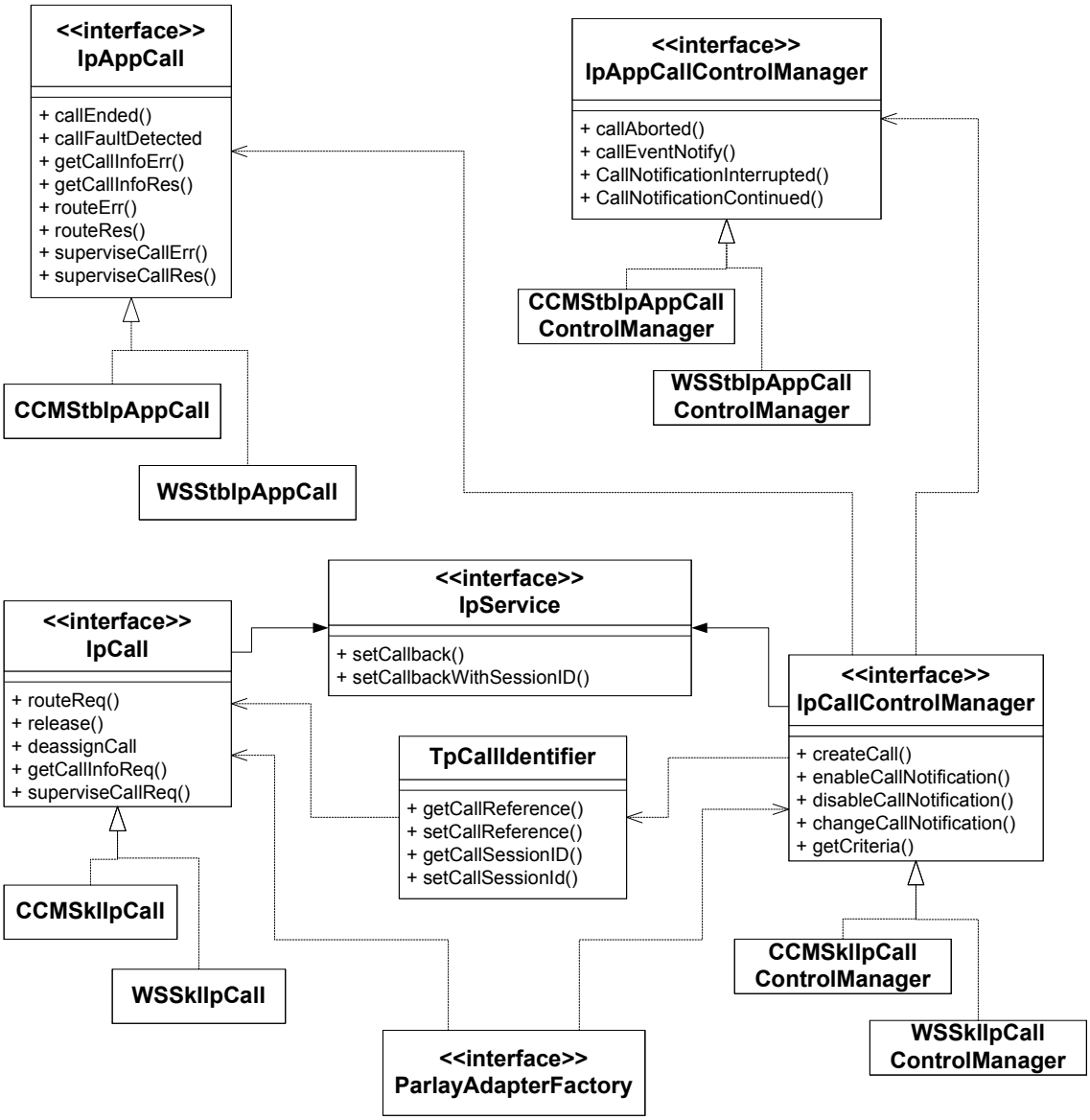


Figure 3-32: Parlay Adapter design

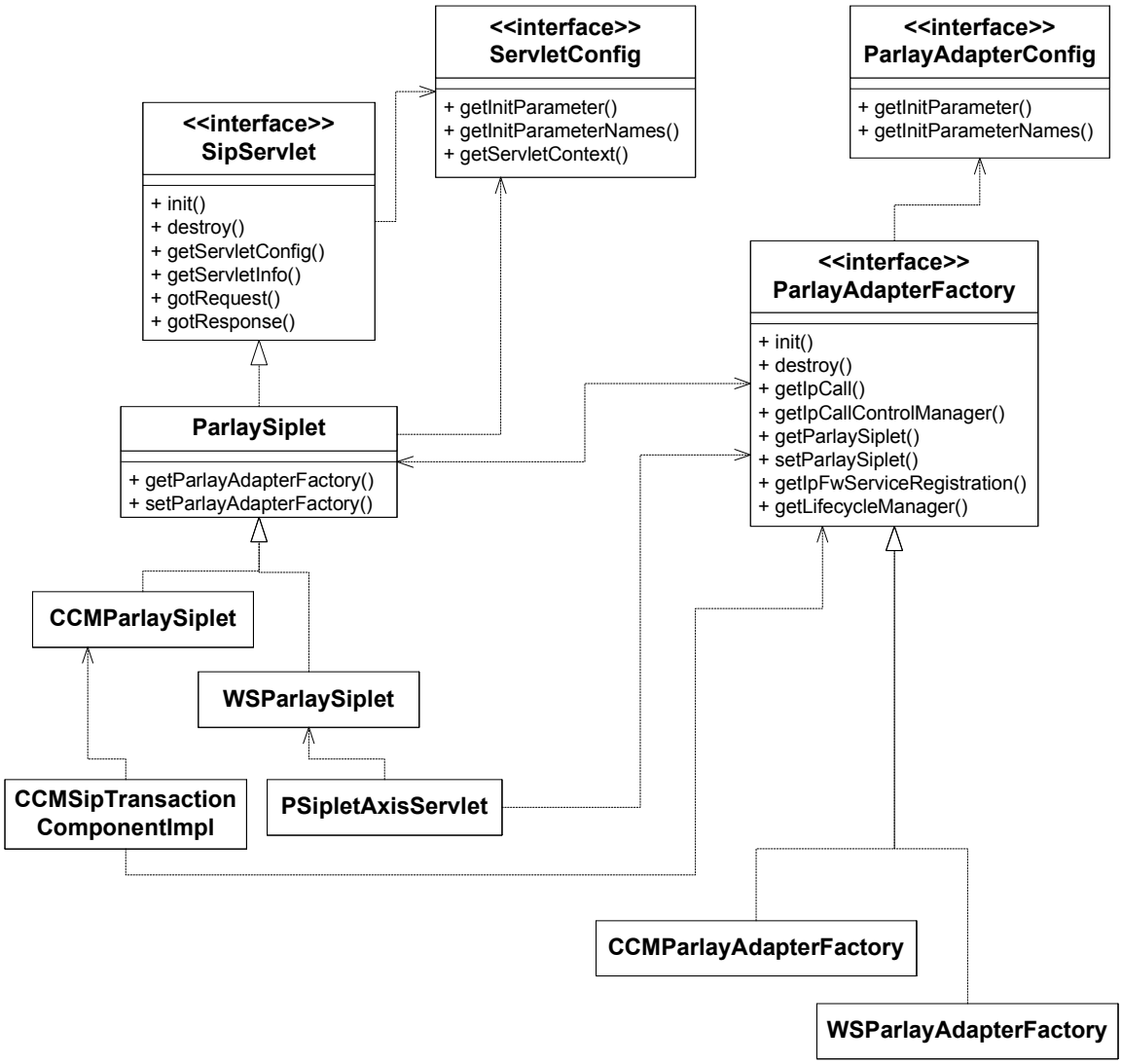


Figure 3-33: Parlay Adapter Factory design

The Abstract Factory pattern (see Figure 3-33) has the following benefits and liabilities [Patterns]:

- It isolates concrete classes and helps to control the classes of objects that an application creates. Because a factory encapsulates the responsibility and the process of creating product objects, it isolates clients from implementation classes. Clients manipulate instances through their abstract interfaces. Product class names are isolated in the implementation of the concrete factory; they do not appear in client code.
- It makes exchanging product families easy. The class of a concrete factory appears only once in an application—that is, where it is instantiated. This makes it easy to change the concrete factory an application uses. It can use different product configurations simply by changing the concrete factory. Because an abstract factory creates a complete family of products, the whole product family changes at once. In the FTW-SP case, a switch from CORBA components to Web services is possible by simply switching the corresponding factory objects and recreating the interface.
- It promotes consistency among products. When product objects in a family are designed to work together, it is important that an application use objects from only one family at a time.

- Supporting new kinds of products is difficult. Extending abstract factories to produce new kinds of Products is not easy. That is because the Abstract Factory interface fixes the set of products that can be created. Supporting new kinds of products requires extending the factory interface, which involves changing the Abstract Factory class and all of its subclasses.

The Abstract Factory pattern implementation in Java uses a technique called *double dispatching* [Eckel TIP]. Java can only perform single dispatching. That is, if Java performs an operation on more than one object whose type is unknown, the dynamic binding mechanism on only one of those types will be invoked. In order to determine more types dynamically, one method call has to be done for each type involved. This is because polymorphism can only occur via method calls. The Abstract Factory pattern consists of two type hierarchies, one for the factory classes and one for the product classes. The first call to the abstract factory determines the type of the factory and the second call to the object creation method in the factory determines the type of the product object.

3.5.6.3 Parlay Callbacks

The classes of the Parlay APIs are strictly organized in pairs. Nne class on the application side is connected to exactly one class on the service side. Figure 3-34 shows the relationship between application objects and service objects of the Parlay Generic Call Control service. third party call control is an interaction where both peers (application and service) act as a client and as a service at the same time. That is, application and service side have to implement interfaces that are called by the opposite object instance asynchronously in the case of certain call events. The `IpCallControlManager` interface is used by the application to create calls or to subscribe to call event reports. The `IpAppCallControlManager` interface is used by the service to notify the application about call events. A call, modeled by the `IpCall` and the `IpAppCall` interfaces, is either created by the application, a process call third party call set-up, or represents an already existing call, that has triggered a notification event. Manipulation of call states by the application is done via the `IpCall` interface, while notification of call events by the service uses the `IpAppCall` interface.

Call control deals with the manipulation of call states and is therefore a state-full service. Asynchronous requests and notifications on one side have to be dispatched to the correct service instance on the other side, representing the targeted call session. The Parlay APIs seem to solve this problem in quite a redundant fashion, by first assigning session IDs and assignment IDs for all state-full operations and additionally to identify object instances by object refernces.

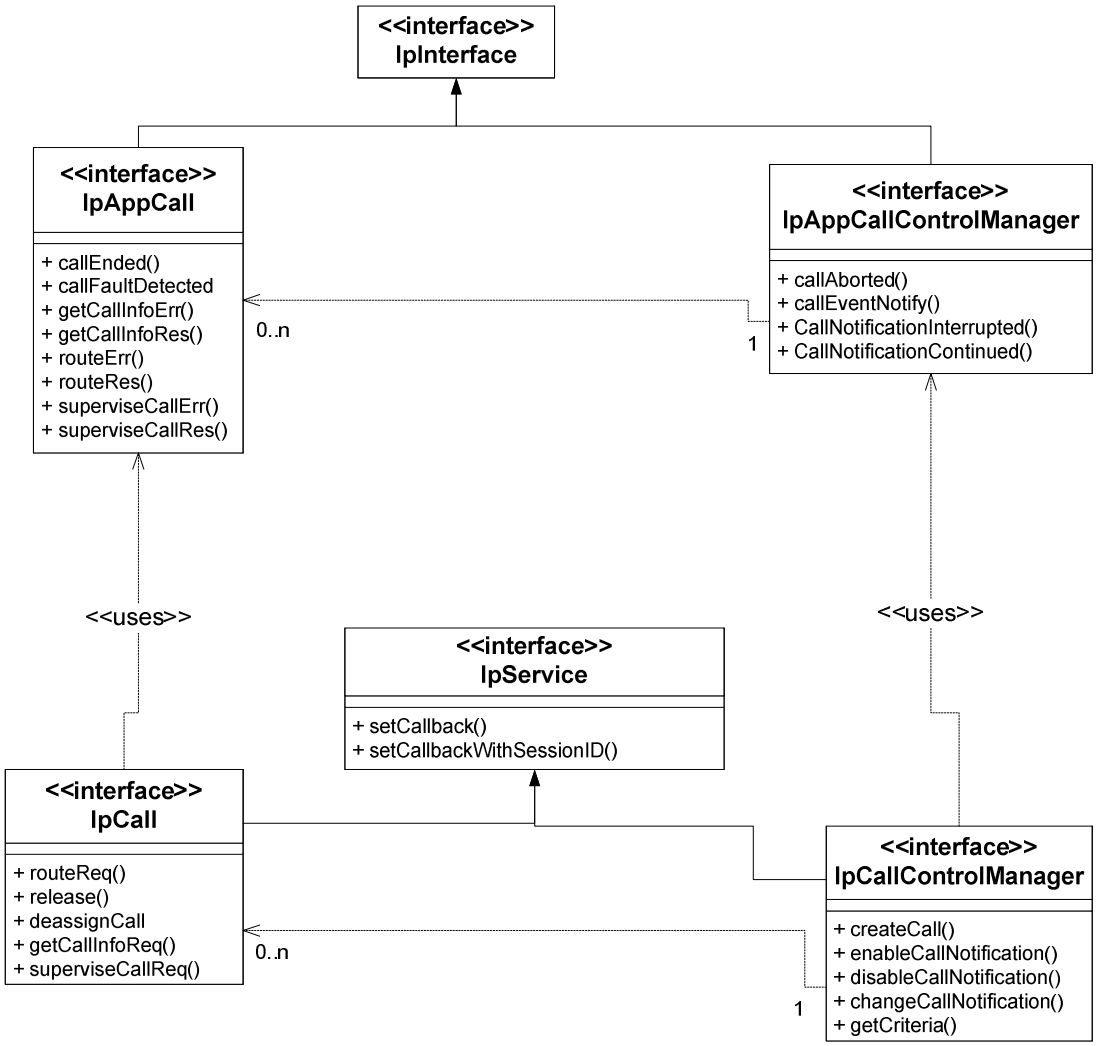


Figure 3-34 Parlay Callback Pattern

This leads to the ambiguous situation, that an `IpCall` service instance can be identified by the TpCallSessionID in the request or an object reference header in the SOAP header part.
The FTW-SP Web Service design therefore modelled all service manager instances as Singletons, so that the service instance of a manager is directly identified by the service URL. Service instances are identified by assigned ID tokens, and object references are derived from those ID tokens. An `IpCall` object instance is for example identified by the call session ID and the `IpCallControlManager` uses a static `HashMap` to reference the call instance by its session ID. Figure 3-35 shows a sequence diagram of a call set-up process in the FTW-SP implementation. In case of a third party call set-up, the application uses a remote call to the `createCall()` method of the `IpCallControlManagerBindingImpl` object, that implements the service skeleton of the call control manager Web Service. The `IpCallControlManagerBindingImpl` object delegates the method call to the ParlayAdapter (`WSSklIpCallControlManager`), which wraps the call control logic of the `PCallManager`. The `PCallManager` instantiates a `PCall`, which represents a SIP call in the Parlay siplet. This call is assigned a unique call session ID, which is used to instantiate an object implementing the `IpCall` interface of the ParlayAdapter (`WSSklIpCall`) and the object reference of this implementing object is stored in a `HashMap` of the `IpCallBindingImpl` with the call session ID as key. Furthermore the call session ID is used to generate

a Parlay object reference by concatenating the `IpCall` service URL and the call session ID. Thus the implementation avoids ambiguities by equating the Parlay object reference with the call session ID and realizes the look-up of the implementing service object instance using the session ID as hash key.

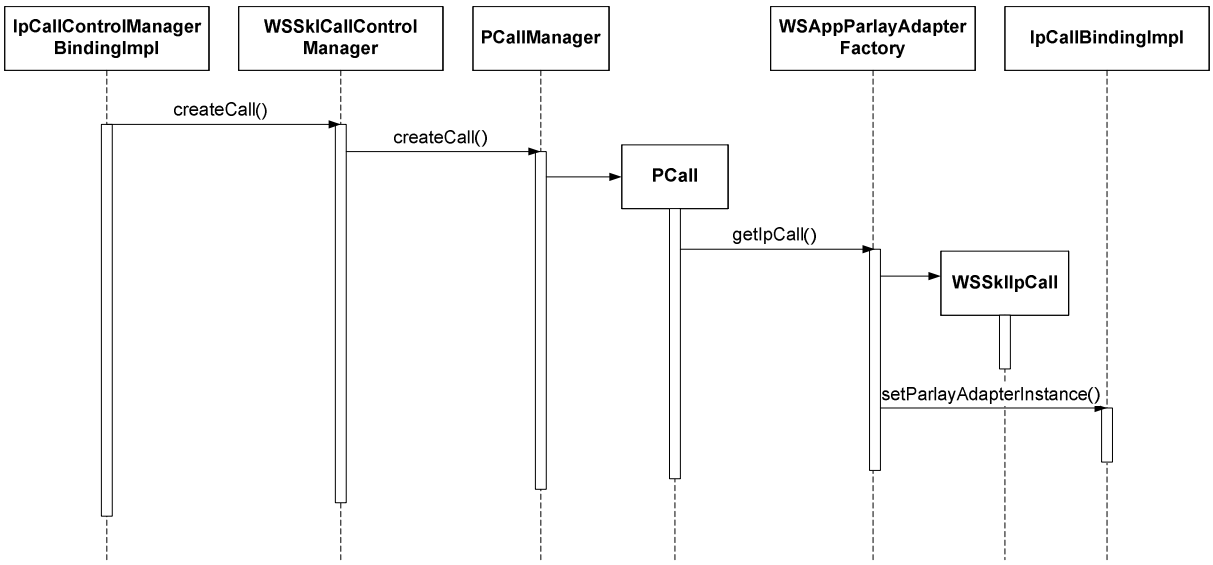


Figure 3-35: Third Party Call Set-Up

The CCM implementation of the FTW-SP implements the Parlay APIs according to the IDL specifications. This specification maps Parlay object references to CORBA object references. This step seems logical and necessary, as a distributed object system cannot work without object references. There is however the risk that an object reference does not match with the call session ID, which is not enforced by syntax and semantics of the API.
The Parlay call control APIs show, that a service oriented architecture does not need the abstraction of an object reference and that the handling of object references can even introduce unwanted complexity and ambiguities in a design.
The Parlay group has realized the problems of the approach to publish Web Service specifications by generating WSDL from a design targeted at distributed object technology. The Parlay X specification [ParlayX] was a reaction on the identified shortcomings and represents a service oriented collection of high level Web Service interfaces to network resources.

3.5.7 A Critical Review on Service References

The previous sections elaborated on the differences between object oriented and service oriented distributed systems and discussed the design implications of a service oriented architecture.
SOAP based Web services are a concrete implementation of SOA principles. A Web Service is specified by its service description, usually in WSDL, covering data types, interface signatures, protocol bindings and a service name and URL. It is very important to understand, that Web services are inherent stateless and thus service URLs cannot be compared to an object reference in a distributed object system like for example a `corbaloc:` URL. A service URL does not identify a specific service instance holding a service instance specific state, but references only an endpoint that is able to perform the service specified in the WSDL. Any state information that is necessary to process a received XML document has to be contained in the document itself, either in the SOAP body part or in the SOAP header part.

The World Wide Web is built on stateless HTTP message interactions. State information is distributed among the peers of the communications (e.g., in the form of session cookies). Moving state from the network to the communication endpoints is a main principle for keeping the Web stable and one of the reasons why the WWW scales to billions of subscribers. Web services apply this concept as well and one may say that they are named due to this principle.
URLs are however somewhat restricted in their use as service references. Especially in the context of event based notification systems, the notion of a service reference becomes crucial [Vinoski 04]. For various reasons URLs do not comply to all requirements of event based systems:

- A URL specifies only a single protocol, but a service could be reachable via multiple protocols. For example, a service might accept messages over both HTTP and SMTP, but any URL for the service can specify only one of those access methods.
- URLs cannot adequately describe some transport mechanism types. For example, message queues typically are defined by a series of parameters that describe the queue name, queue manager, get and put options, message-expiration settings, and message characteristics. It is not practical to describe all of this information in some form of message-queue URL.
- URLs do not necessarily convey interface information. This is especially true of HTTP URLs, because Web services generally tunnel SOAP over HTTP. Given a URL there is no indication whether the resource it represents is a Web page or a Web service. Many Web services development kits automatically return the service's WSDL definition in response to an HTTP GET on the service URL, but that is only a convention, not a standard.

One proposal to deal with these issues is the Web services Addressing (WS-Addressing) specification [WS-Adr] by BEA Systems, IBM and Microsoft. WS-Addressing provides transport-neutral mechanisms to address Web services and messages. The specification defines XML elements to identify Web services endpoints and to secure end-to-end endpoint identification in messages. It enables messaging systems to support message transmission through networks that include processing nodes such as endpoint managers, firewalls, and gateways in a transport-neutral manner. WS-Addressing defines two interoperable constructs, endpoint references and message information headers, that convey information that is typically provided by transport protocols and messaging systems.
It remains however to be seen how such proprietary proposals are accepted by the Web Service community. Care has to be taken that service reference proposals do not break the principles postulated for a service oriented architecture.

3.5.8 Performance in Distributed Service Platforms

Distributed systems have a number of architectural challenges [WS Arch]:

- Problems introduced by latency and unreliability of the underlying transport.
- The lack of shared memory between the caller and object.
- The numerous problems introduced by partial failure scenarios.
- The challenges of concurrent access to remote resources.
- The fragility of distributed systems if incompatible updates are introduced to any participant.

These challenges apply irrespective of whether the distributed object system is implemented using COM/CORBA or Web services technologies.
The Parlay APIs define interfaces to remote objects in order to allow applications in an „untrusted“ application domain to control resources (e.g., a call control server) in a „trusted“ network provider domain. Some kind of distributed system technology has to be used to implement communication between service objects and application objects. This thesis argues that a service oriented system architecture based on SOA and Web services matches much better the requirements of large scale communications systems than distributed object technologies like CORBA.
When designing a distributed communication system, the performance implications of ‘remoteness’ have to be considered in the system design.
For a design based on the Parlay APIs is has to be pointed out that components that implement Parlay interfaces can but have not to be remote. Typically the round trip time (RTT) of a remote service component invocation is several orders of magnitude larger than a local method call and depends on network conditions like network capacity, delay, or error rates.
Remote service component interfaces however make it possible to deploy different service components on different physical machines, to access service components in different domains and for load balancing inside a single domain.
Figure 3-36 shows a simple and a large scale deployment for a system based on the Parlay APIs. The large scale deployment shows that by distributing service components to different hardware platforms, framework, management and business logic tasks can be scaled to the needed performance. Furthermore the picture suggests that network technology independence may be realized by working with an abstract call leg service component that actually runs on different hardware modules specialized for a particular network protocol.

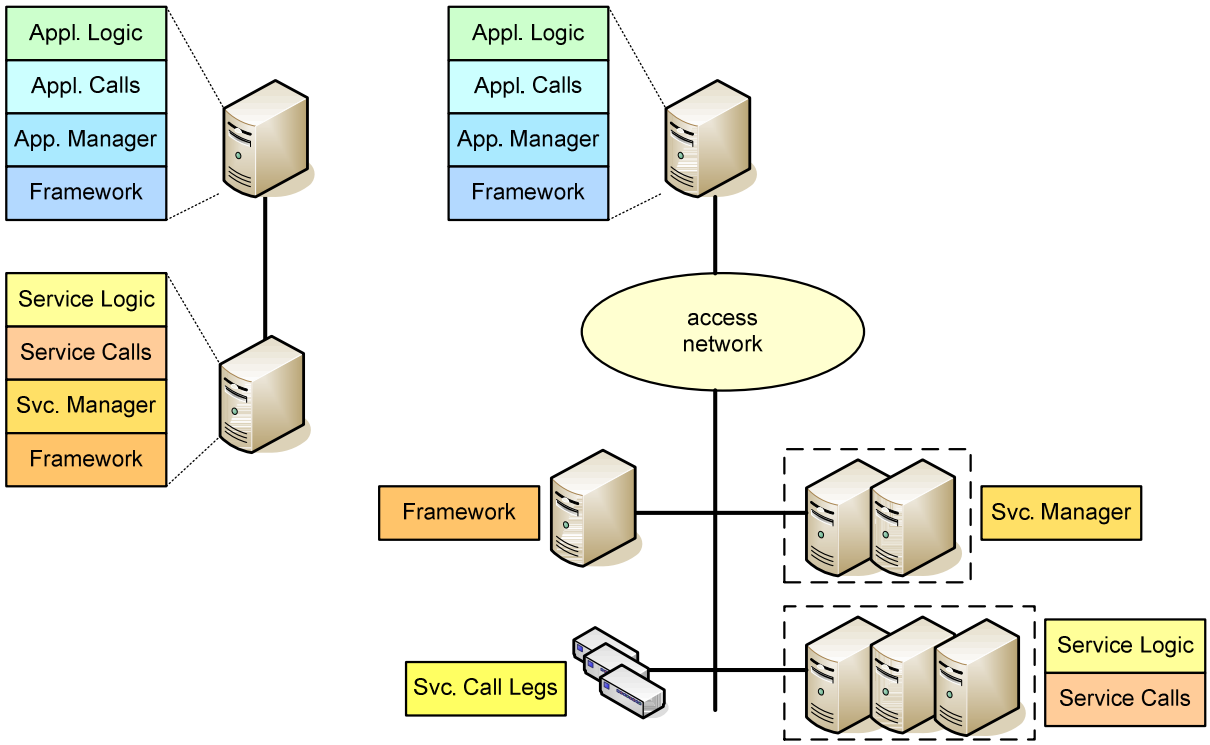


Figure 3-36: Simple and large scale deployment scenario for a Parlay system

In order to compare the performance of Web services and CORBA based service components, the call control service of the FTW SP was designed to be independent of the technology used by an application for remote access. Thus the Parlay call control SIP servlet was either accessible via CORBA components or via SOAP Web services. Abstraction of the remote access transport technology was done by introducing a set of classes that wrapped the Parlay interfaces, called Parlay Adapter (see *Parlay Adapter Design of the FTW Service Platform*).
For performance measurements a SIP traffic generator was implemented called *Sim Siplet* in form of a SIP servlet (see Figure 3-37).

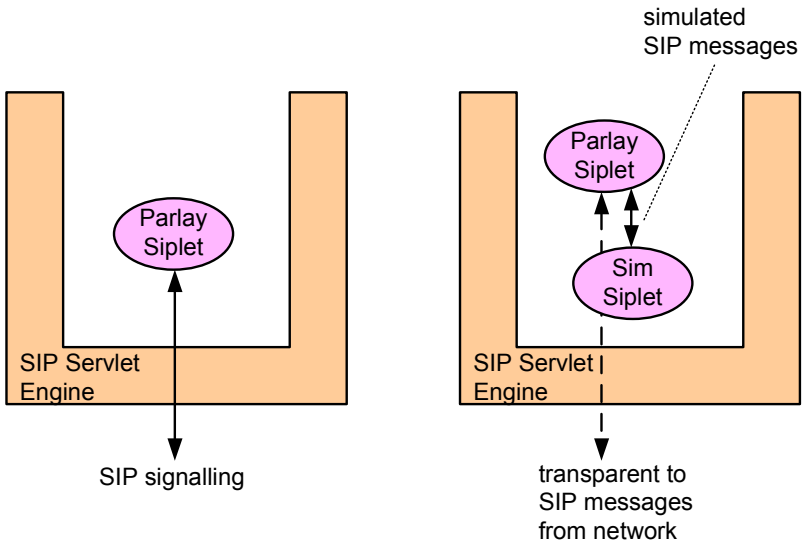


Figure 3-37: SIP traffic generator Sim Siplet implemented as a SIP servlet

The Sim Siplet generates calls according to a Poisson distribution with a mean call rate of λ. The random variable X is the number of calls generated per time interval and the parameter λ is the expected value (mean) of calls per time interval. The probability mass function of the Poisson distribution is

$$f(k;\lambda) = \frac{e^{-\lambda}\lambda^{k}}{k!}$$

Alerting duration and call duration are modelled according to a negative-exponential distribution. The parameter 1/μ is the expected value of the alerting/call duration. The cumulative distribution function for the interarrival time is

$$F(t;\mu) = 1 - e^{-\mu t}$$

A Mersenne Twister Random Number generator with a period of 10^{6001} has been used. The CERN Open Source Libraries for High Performance Scientific and Technical Computing in Java were used in the implementation.

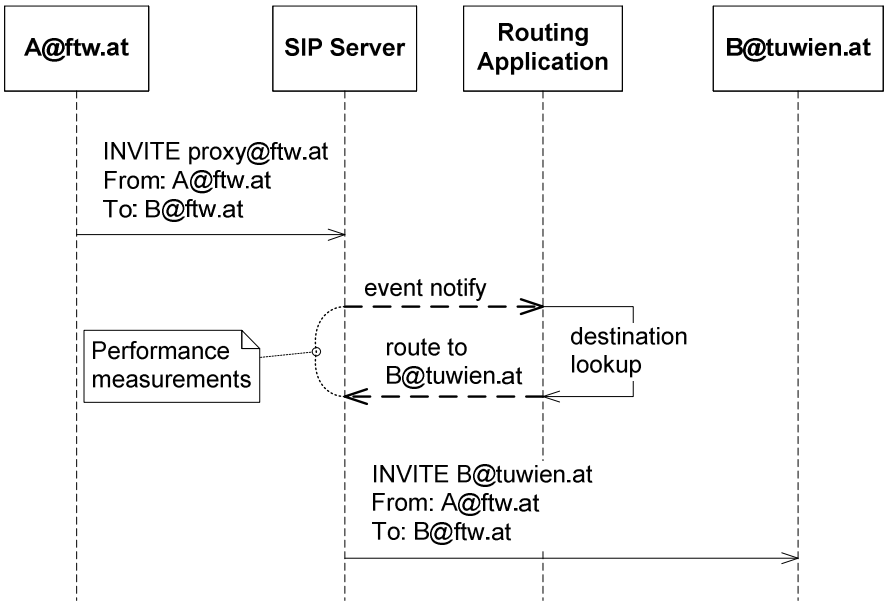


Figure 3-38: Call scenario for performance measurements.

Figure 3-38 shows the call scenario that was used for performance measurements. A call set-up from the SIP protocol stack triggers an event notification to a routing application. The read arrows indicate remote access to service components (either via SOAP or CORBA).

Time t_{RMI} of a remote eventNotify() method call and of a remote routeRes() method call are measured at the server. The time t_{app} the application needs to compute a response is measured at the application. The result $t_{res} = t_{RMI} - t_{app}$ is the time that is necessary for the used technology (either CORBA or SOAP) to send one request from the server to the application and get a result back.

In order to get a good resolution of the result times, the Win32 timer function GetPerformanceCounter() with a resolution of 50 µsec was used via the Java Native Interface. The Java standard timer System.currentMillis() has only a 10 msec resolution on Windows 32 systems.

The Server (1300 MHz Pentium, 260 MB Ram) and the Application (500 MHz Pentium, 260 MB Ram) were connected via a 100Mbit Ethernet LAN. Calls were generated with a mean rate λ of 0.5 calls/sec and a mean call duration $1/\mu$ of 10 sec. Table 3-3 shows the parameters of the service component invocation times t_{res} for CORBA and SOAP.

t_{res}	CORBA	SOAP
mean	3.8	223.1
median	2.7	172.5
std. deviation	5.6	95.7
skew	18.4	2.6
curtosis	437.9	9.0
10%-90% quantile	2.5	172.5
25%-75% quantile	0.6	96.8

Table 3-3: Invocation time comparison for SOAP and CORBA

This exemplary performance comparison between service component invocations based on SOAP Web services and CORBA components shows, that using Web services involves a sub-

stantial additional delay. The processing delay for Web services is caused by the processing of a Web service request in the service component. First the HTTP message that contains the SOAP message in the body has to be received by the HTTP servlet container (Apache Tomcat). Then the SOAP XML document has to be parsed by the Web services framework (Apache Axis). Finally the Web services framework has to instantiate Java objects that represent the SOAP message parts.
The invocation times for Web services can certainly be improved by using Web servers and XML parsers with better performance characteristics. Nevertheless a pure binary protocol like CORBA will always have performance advantages against an XML document based protocol. These results show that the implications of using Web services for remote invocation of a service component have to be understood well and considered in system designs.
It is the general experience of the author however, that the achievable performance of Web service technology is sufficient for a major part of applications and services in next generation telecommunication networks. If higher performance is indeed necessary, it has to be emphasized that Web services bindings to other protocols than HTTP (e.g., for the Java Message Service - JMS) are available.

3.6 Softswitch Architectures based on Parlay

In the last years the convergence between circuit switched and packet switched communication networks has become more and more important, not at least because of the growing Voice over IP or rather Multimedia over IP market. Currently, network providers from the classical circuit switched world have to think about enabling interconnection to the IP domain over appropriated gateways. Another important issue is the separation of service logic from the network technology in order to make services usable from different types of networks. If this separation is achieved over open interfaces, so called third Party Service Providers could offer their services in a most flexible way. Hybrid Components, called softswitches, can fulfil both, gateway functionality and access to network independent business logic using standardized interfaces. One possibility for such an interface could be the Parlay APIs. In the following three different softswitch architectures are proposed (see also [Stadler]) that support connectivity to the PSTN as well as to the IP world. They differ in the placement of network specific components and in the way of gateway control. In each case Parlay is used between the networks specific Call Control and the independent application.

3.6.1 A Flexible Softswitch Architecture

Due to the growing importance of packet switched technology in contemporary communication networks, there is a promising tendency of convergence between the world of telecommunication and that of data communication. As much as the transport of voice or more general multimedia data over packet switched networks provides an enormous opportunity, it also implies incalculable risks for current circuit switched network providers.
Beside their PSTN services as main business, they have to decide about the development of new packet switched networks, in many cases from the scratch. Huge investments would have to be made for hardware and software, new groups of users have to be addressed and new, more powerful services have to be invented to get back the high costs.
To reduce the financial effort, it would be a goal to utilize equipment, which is applicable for both domains. Furthermore, services should have to be developed only once. Obviously this reduces the cost of the creation of such services and moreover it enables a new class of applications, based on the interaction of different types of communication networks.
So it seems possible that for the years of migration, systems for hybrid network handling could be a meaningful alternative. Using current architectures, which are known from PSTN switches, this is not easily possible. In most cases such devices are designed in a more or less monolithic manner. The software is tightly coupled to the hardware mostly, interfaces are rarely standardized and applications are hardly reusable.
This implies two further problems. Because of the lack of an open service interface, only the switch developers are able to develop applications for a certain piece of hardware. This affects the spectrum of available services and increases the time to market. The second issue is that it is impossible to offer network capabilities to so called third party service providers. So only network providers have the ability to act as service providers. Figure 3-39 shows this conventional architectural approach.

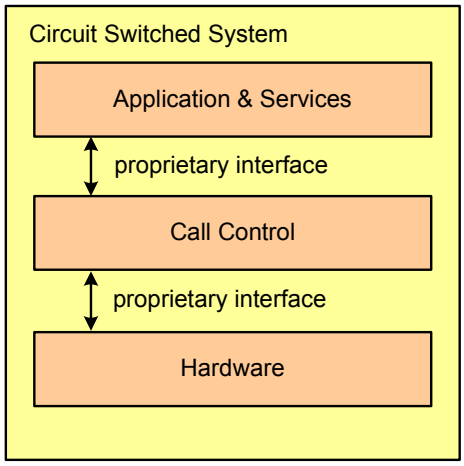


Figure 3-39: Classical Circuit Switched Architecture

Above considerations motivate the redesign towards a more flexible and more open architecture. Figure 3-40 depicts such an approach, which is commonly called a softswitch.

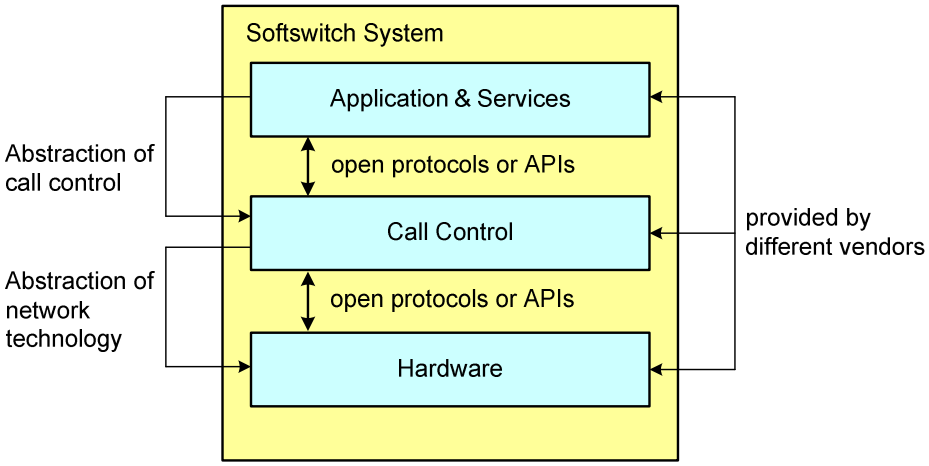


Figure 3-40: Softswitch Architecture Approach

A softswitch enables the combination of applications, call control and hardware from different vendors as well as an abstraction of the applied hardware to the call control layer and of the communication network to the applications, by the usage of open interfaces.

3.6.2 Advantages of using the Parlay APIs in a Softswitch architecture

The Parlay APIs have initially been proposed for providing third party service developers with a secure interface towards telecommunication resources of a network service provider. Towards the application program only a highly abstracted, object oriented and fully distributed call-processing model is visible. Existing solutions usually offer access to one specific switching resource (e.g., an IN Switching Control Point – SCP), where the calls and call legs are abstractions of the same network technology. A call that crosses networks boundaries is therefore only visible up to the gateway where the call leaves the Parlay controlled domain.
This thesis argues that this classic Parlay usage model fits very well in the proposed Softswitch architecture where the call control layer is separated from the application and service logic layer by open and standardized interfaces. The Parlay APIs facilitate the implementation of Softswitch architectures by a mature security and management concept, by an object-oriented design and by the distributed nature of all of the interfaces.

It is proposed that apart from this classic 'network technology centric' usage model the Parlay APIs can be used to present a call model towards the application logic where a single instance of a call object can have call legs that use different access technologies. The Parlay Multi-Party call model defines the call leg as an abstraction of a signalling relationship between an end-point and a call object. Call legs are modelled as independent interfaces that are implemented by distributed objects (e.g., by using CORBA technology). This means that the Parlay specification allows call legs that belong to one call to be implemented by objects that run on different physical machines. A network resource that offers access through some specific technology (e.g., circuit switched ISDN, analogue lines, SIP, H.323) can host a Parlay call leg that represents and controls this half of the call. The call legs are associated with a call that runs for example on a separate call-processing server. In this architecture a signalling gateway between different network technologies is no longer necessary as its function is implicitly carried out by a common Parlay call processing model. Thus, call set-up procedures and call control tasks are unified and simplified through a common and distributed call model.

3.6.3 Media Gateway Control

Telephony gateways are network elements between circuit switched (e.g., Public Switched Telephone Network (PSTN)) and packet switched networks (e.g., Internet) that basically perform two types of tasks: signalling translation and media translation. Examples for such telephony gateways were traditional telephony switches, where the call processing (signalling) and switching (media transport) functionalities were typically bundled within the same physical platform. The European Telecommunications Standards Institute (ETSI) project Telecommunications and Internet Protocol Harmonization over Networks (TIPHON) undertook the effort to identify technical agreements required for the interoperability of telephony gateways. The result of this work is a decomposition model for telephony gateways, where the gateway is divided into three entities – a Signalling Gateway (SG), a Media Gateway (MGW) and a Media Gateway Controller (MGC). Nevertheless, the decomposed gateway appears to the outside as a single gateway. The gateway decomposition does not necessarily imply a physical breakdown, but usually media translation requires more hardware resources than signalling translation. The goal of the logical separation is also to allow the signalling that enables services to be normalized into a media control interface, thereby separating services from the underlying media handling. The key advantages of this decomposition approach are the higher degree of scalability and maintainability the resulting gateway solutions provide and the opportunity to introduce new services more quickly. [TIPHON]
Each of the three entities of a decomposed gateway has to perform different tasks. The SG mediates between circuit switched (e.g., Signalling System 7 (SS7) or Integrated Services Digital Network [ISDN]) and packet switched (e.g., Session Initiation Protocol [SIP] or H.323 [H.323]) signalling protocols. The MGW focuses on the audio signal translation function, performing conversion between the audio signals carried on circuit switched networks and data packets carried over packet switched networks. MGWs can be realized as Voice over Internet Protocol (VoIP) gateways, Voice over Asynchronous Transfer Mode (VoATM) gateways, modem banks, cable modems, set-top boxes or Private Branch Exchanges (PBX). Finally, the MGC acts as a manager within the telephony gateway and instructs the MGW according to the signalling information received by interfacing with the circuit switched network via the SG on the one side and with the packet switched network via a signalling device such as a SIP server or an H.323 gatekeeper on the other side. By allowing one MGC to in-

struct several MGWs the distributed architecture provides the means to control a vast number of calls simultaneously.
The Media Gateway Control Protocol [MGCP] serves as an internal master/slave protocol between the MGC (master) and the MGW (slave) of an ETSI/TIPHON compliant decomposed telephony gateway. MGCP retains simplicity by utilizing an Application Programming Interface (API) that is made up of a fairly small set of eight commands, which are grouped into three basic command sets: device management, connection control and in-band signalling handling. Commands may take numerous arguments and are transmitted using plain American Standard Code for Information Interchange (ASCII) text. Like SIP and H.323, MGCP also relies on already established standards such as the User Datagram Protocol [UDP] for transmitting messages between the MGC and the MGW (with the expectation that transmission reliability will be managed by the specific implementation), the Session Description Protocol [SDP] for negotiating the media aspects of a call (e.g., ports and Internet Protocol [IP] addresses, voice coding methods and other connection type parameters), the Real-time Transport Protocol [RTP] used by the MGW for handling the transport of media streams, and the Internet Protocol Security [IPSEC] protocol for providing secure connections.
The main entities of the connection model used in MGCP are Endpoints and Connections. Basic constructs of Endpoints and Connections are used to establish voice paths between call participants. Endpoints are sources or sinks of data and can be either physical or virtual. Physical endpoint creation requires hardware installation (e.g., an interface on a gateway that terminates a trunk connected to a PSTN switch), while software is sufficient for creating a virtual endpoint (e.g., an audio source in an audio-content server). Connections may be either point-to-point or multi-point, whereas one or more connections are required to set up a call. A point-to-point connection associates two endpoints. Once this association is established for both endpoints, data transfer between these endpoints can begin. A multi-point connection is established by connecting the endpoint to a multi-point session. Connections can be established over several types of bearer networks (e.g., Transmission Control Protocol (TCP) [TCP] / IP networks or Asynchronous Transfer Mode (ATM) networks). For both, point-to-point and multi-point connections the endpoints can be in separate gateways or in the same gateway.
The control primitives for operations in MGCP are Signals (sent from the MGC to the MGW) and Events (sent from the MGW to the MGC). The concept of Signals and Events is used for establishing and tearing down calls. Operations are performed by applying Signals (e.g., 'ringing' or 'busy') to endpoints and detecting Events (e.g., 'off-hook transition' or Dual Tone Multi Frequency (DTMF) tones) from endpoints. The Signals and Events that are needed to support a specific telephony function or type of endpoint are grouped into an abstraction called package. Example packages defined in the MGCP specification include: Generic Media Package, DTMF Package, Trunk Package, Handset Emulation Package, and RTP Package.
The benefits of MGCP are manifold: the potential to control very large deployed systems, the opportunity of simple integration into SS7 networks, which gives greater control and throughput in handling calls and finally the fact that gateways utilizing MGCP can co-exist in networks with other SIP or H.323 compliant entities. One of the weaknesses of MGCP is the time-consuming implementation of the protocol for smaller applications.
MGCP is based on concepts originally developed in the telephone industry for Advanced Intelligent Networks (AIN), which assume that the logic for providing services is centralized. The Internet and other IP networks focus on intelligence at the edges of the network. This contradistinction was amongst the reasons why the evolution of the MGCP specification was

largely influenced by 'political' conflicts between proponents of alternative proposals for decomposed gateway architectures. In particular, MGCP owes its origin to the confluence of the Simple Gateway Control Protocol [SGCP] and the Internet Protocol Device Control [IPDC] protocol. Nevertheless, MGCP is not the final word on gateway control. The IETF combined forces with the ITU to endorse the MEGACO/H.248 [MEGACO] standard in late 2000. MEGACO/H.248 draws heavily from MGCP plus introduces several enhancements, as for instance: support for multimedia and multi-point conferencing enhanced services, improved syntax for more efficient semantic message processing, TCP and UDP message transport options, text or binary encoding and an expanded definition of packages. Accordingly, MEGACO/H.248 can be considered as a richer approach to gateway control, nevertheless implementation is a lot more complex.

3.6.4 Parlay Based Softswitch Architectures

In this section the following three softswitch architectures are compared:

- Softswitch with common generic call control
- Softswitch with distributed call control
- Softswitch with distributed call control using an external gateway (Pseudo Softswitch)

The same scenario will be used to illustrate the different architectures. User A that is located in the PSTN wants to call user B, who has configured a call routing application that forwards the call into the IP domain, or more specifically to user B's current SIP address.

The application resides in the service provider's domain and offers two access points. First the application has a Web interface that enables the user to configure his routing application via HTTP, for instance to set a SIP URI where a call to his PSTN number should be forwarded to. Second the application implements the Parlay APIs that are used by the call control components to access the application, e.g., for the purpose of address translation or for the interaction towards the media gateway controller.

Call control components are separated in two parts, the generic and the specific call control part. The generic call control part represents the call towards the application, but is independent of the underlying signalling network. It interacts with the specific call control, which represents a certain call leg towards the protocol stack of the corresponding network via an open interface. In our scenario the ISUP and SIP protocol stacks have to be implemented by the specific call control instances.

The media gateway controller is either invoked by the application itself using the Parlay interface, or via the generic call control part. It controls the media gateway, which actually is responsible for media conversion between the involved networks and deals with the appropriated transport protocols or mechanisms.

The proposed architectures are analysed by comparing the degree of complexity in the application versus that in the generic call control. The application shall be independent of the underlying network technology, so that application development does not require deep knowledge of network specific parts of the call control functionality. Furthermore the three proposed softswitche architectures differ in how its components are distributed which has influence on communication overhead. Finally, it is important that each component id modelled as an independent building block, ideally using open interfaces so that components of different vendors can be combined.

3.6.4.1 Softswitch with common call control

The softswitch with common call control (see Figure 3-41) only needs one Generic Call Control (GCC) component. The GCC unit is connected to on the network side with different protocol specific Call Control (CC) units and on the application side with the routing application. The interface between the GCC component and the application is the Parlay GCC interface. The application obtains a reference to the GCC service instance from the Parlay framework APIs.

The GCC component has to maintain protocol specific state for every CC unit involved in a call, which increases complexity. The GCC component represents the call towards the application while the single call legs are maintained by the specific CC components. This results in a distributed management of call states.

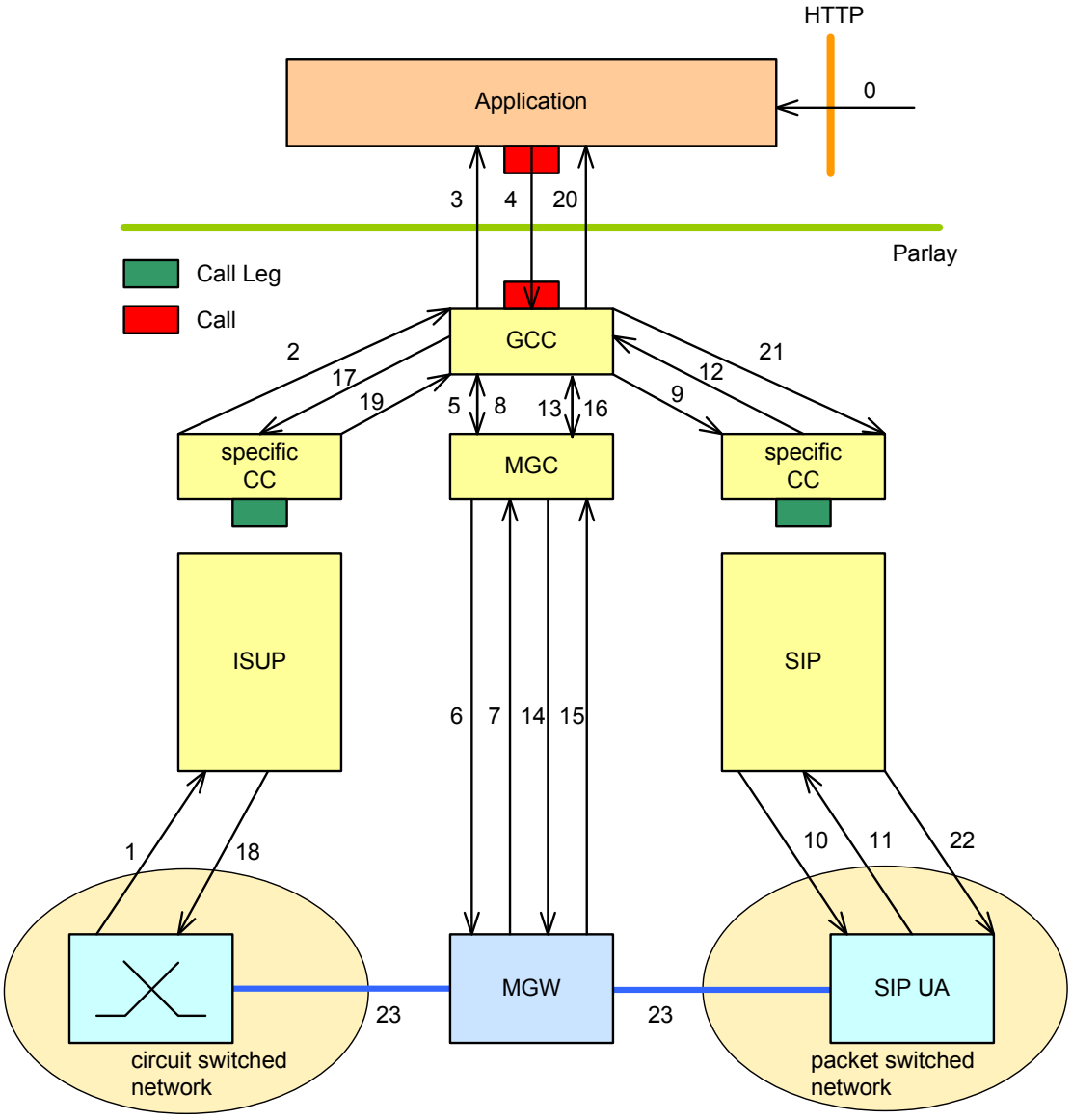


Figure 3-41: Call setup scenario with a common Generic Call Control component

The message flow in Figure 3-41 for an incoming call from the ISDN side towards the IP domain is now described in detail. For the remaining two architectures only the differences compared to this flow will be described..

At first, after the deployment and setup of the application, the subscribed users are enabled to place some user related configurations, e.g., over HTTP (0). Afterwards, a call-setup request

coming from a PSTN switch is delivered to the ISUP protocol stack of the softswitch (1). It is forwarded to the protocol specific CC entity, which generates a call leg for this call, and further to the GCC unit (2), which creates a call and contacts the application with origination address, destination address and other call related parameters (3). The application finds the new destination address (where the user has defined to receive this call), generates a call representation and sends a route request to the GCC unit containing the new address (4). In the present scenario it is a SIP address. The GCC unit recognizes the type of the address and prepares a PSTN-to-IP call.
For calls between different kinds of networks (e.g., circuit switched and packet networks) the usage of a Media Gateway (MGW) is necessary. The MGW converts the media streams between two networks. Therefore the GCC unit contacts a Media Gateway Controller (MGC), which tries to occupy a trunk between the MGW and the PSTN switch (5-8). If this does not succeed, a busy signal is sent to the PSTN user and the call setup ends. Otherwise the GCC unit contacts the SIP specific CC unit with the previously resolved destination address (9). The SIP specific CC unit creates the second call leg of this call and forwards the call request to the SIP stack.
The SIP stack sends a SIP INVITE request to the destination address (in this case it is the address of a SIP UA) (10). The SIP UA indicates an incoming call. Supposed the user wants to accept this call, the SIP UA sends an OK response to the SIP stack of the softswitch (11). This response is forwarded to the SIP specific CC, which notifies the GCC unit (12). The next step is to inform the MGW about the media settings of the callee via the MGC (13-16).
Afterwards, the GCC unit informs the ISUP specific CC unit that the call-setup was successful (17). This information is forwarded to the ISUP stack and further to the PSTN switch (18). Finally, the ISUP specific CC notifies the GCC unit (19) that appraises the application (20) and the SIP specific CC (21) that the call-setup succeeded. The SIP CC forwards this information to the SIP stack and a SIP ACK request is sent to the SIP UA (22). After completion of the call-setup, media data exchange (voice, video) starts (23).
The major benefit of this architecture is that the application is connected only to one entity (GCC component). Therefore, the developer of the application does not need to care about underlying network issues. Call control is delegated to the common GCC unit. This abstraction simplifies service creation, which reduces time to market. Another important benefit is the low traffic over remote Parlay interfaces which results in shorter call setup times.
As a drawback of this architecture it can be noticed that the GCC unit, protocol specific CC, protocol stack and MGC are connected by proprietary interfaces and a more or less distributed state-machine between the CC entities. This means that the GCC component has rather high complexity and requires customized implementation of specific call control protocols.

3.6.4.2 Softswitch with Distributed Call Control

In the softswitch architecture with distributed call control (see Figure 3-42) there are no proprietary interfaces, due to the usage only of open Parlay APIs between all protocol specific components and the application. Every protocol specific component contains a GCC unit and is therefore able to communicate with the application.
It is important for the PSTN-to-IP scenario that in the distributed call control architecture the GCC unit of the ISUP stack (note that is not a protocol specific CC unit) contacts the application directly (2) over Parlay. The application contacts the MGC that orders the MGW to occupy a trunk to the PSTN switch (3-6). After that, the application performs a call-setup (7-10) into the SIP domain, contacts MGC to configure callee's media settings (11-14) and notifies

the GCC of ISUP stack that the call-setup has succeed (15). A call representation is created in both GCC (ISUP and SIP) and in the application.
Comparing this architecture to the softswitch with common call control, it can be noticed that protocol specific components can be purchased from different vendors, due to open interfaces and that every component has its own state machine.

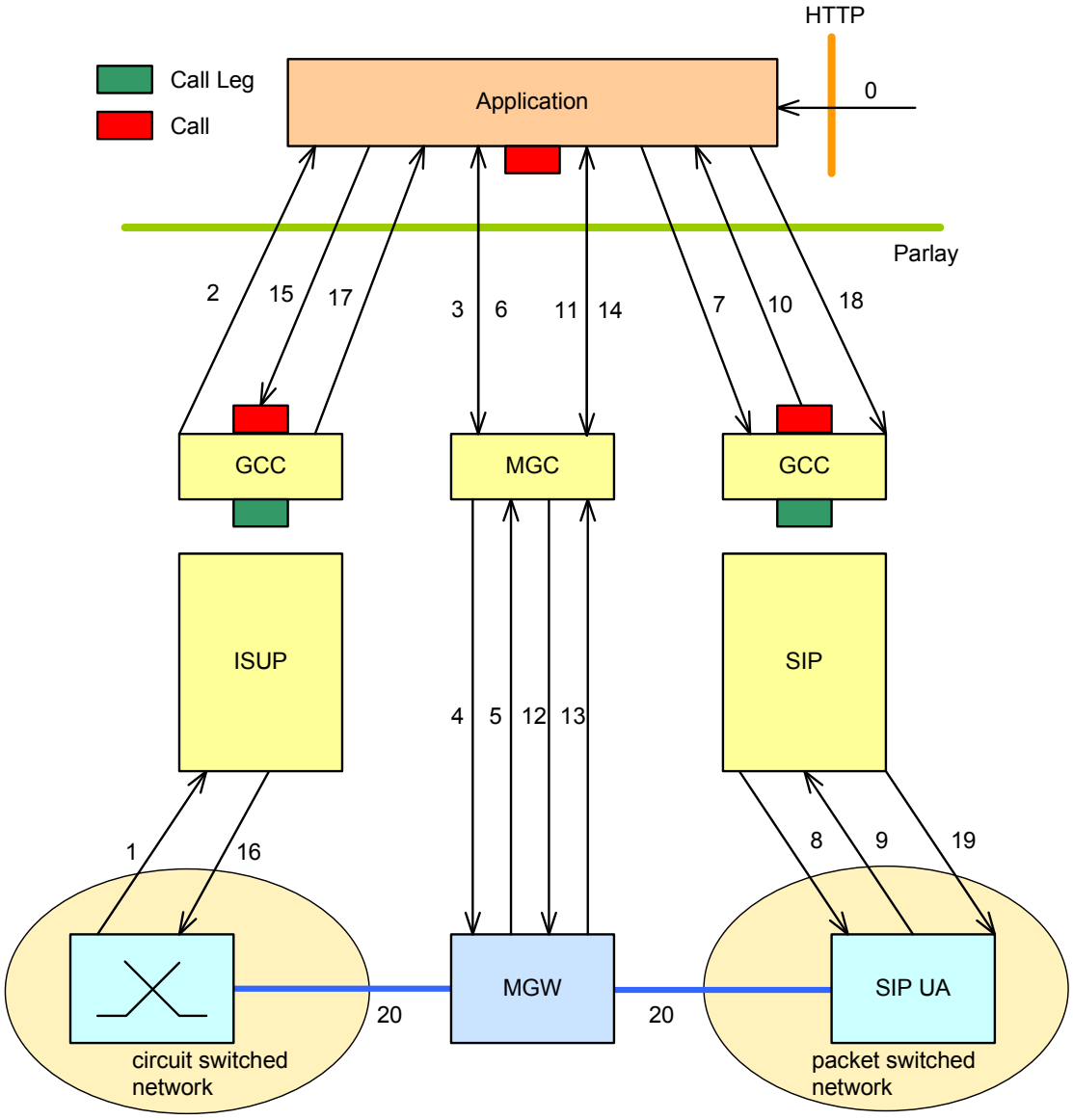


Figure 3-42: Softswitch architecture with distributed call control

The traffic between the application and the GCC components over Parlay however doubles, which will increase call-setup times. Another difference is that complexity has been moved from the GCC to the application. The application has to recognise the type of the call (e.g., IP-to-PSTN call) and to use the corresponding GCC unit. Furthermore, the application has to maintain states for different GCC units in a call.

3.6.4.3 Softswitch with Distributed Call Control and External Gateway

The last architecture that is propose is a softswitch with distributed call control an external MGW. This architecture is a variant of the softswitch with distributed call control. The difference here is that there is no MGC component in the softswitch itself, which demands the usage of an external gateway. Due to this reason this architecture is further on called pseudo-

softswitch. The call-setup is quite different in comparison to the two other architectures (see Figure 3-43).

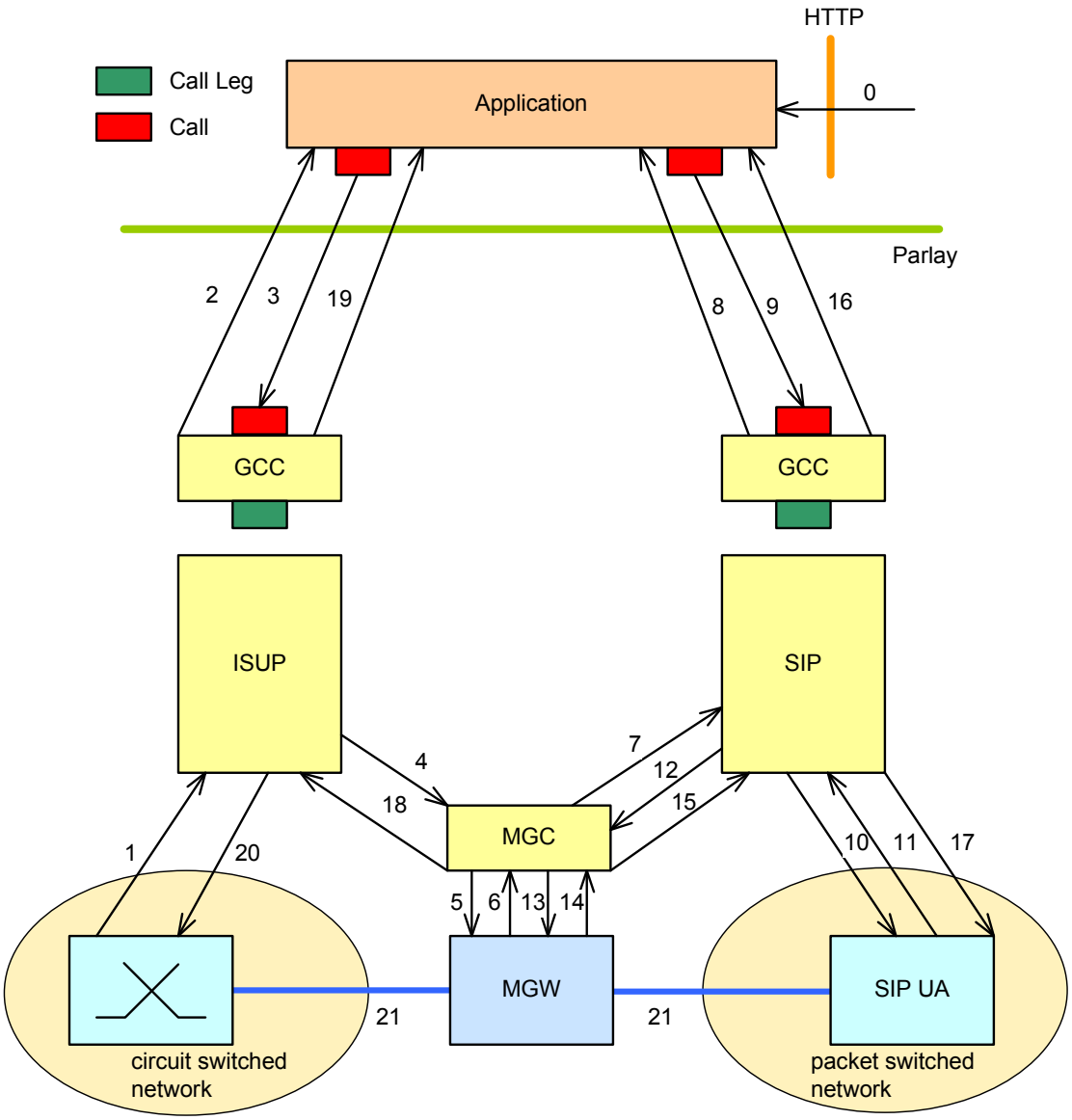


Figure 3-43: Softswitch with distributed call control and external media gateway

After receiving a call-setup request from the ISUP, the GCC of the ISUP side contacts the application (2). The application recognises that this call should be routed to the IP network and forwards it over ISUP stack to the MGC (3, 4). Note that a call is created in the application and in the GCC of the ISUP stack.

The MGC reserves resources from the MGW (5, 6) and sends a SIP INVITE request to the SIP stack of the softswitch (7). Afterwards, the GCC of the SIP side notifies the application that determinates the address of the callee and contacts the SIP GCC (8, 9). Another call is created in the application and in the GCC of the SIP stack. The callee receives a SIP INVITE request and replies with an OK response (10, 11). Now the SIP stack informs the MGW over the MGC (12-15) and notifies the application and the callee that the call-setup has succeeded (16,17). Finally, the MGC reports the call-setup results to ISUP stack, which forwards this report to the application and to the PSTN switch (18-20).

This architecture tries to avoid drawbacks of the previous architecture by reducing the traffic over Parlay and by making the application less complex in terms of call control. There are

two calls in the application now, which can be maintained separately. This makes the application more flexible and reduces the developing time. Furthermore, the application does not need to communicate with the MGC. Protocol specific components are independent from each other and so they could be obtained from different vendors.
One drawback is the usage of an external MGC component.. Furthermore, the application has to perform the mapping between gateway addresses and user addresses.

3.6.5 Conclusion

The world of telecommunication networks is rapidly converging. Network providers, especially circuit switched network providers, are facing the problem that they have to decide about when and how to make the step into the packet switched domain. Such a step might be coupled with huge investments. To reduce these high costs and risks in the periods of migration, more flexible hybrid architectures might be helpful. They should enable the usage in, and the interconnection of, different types of networks in an open manner. Architectures commonly known under the term softswitch fit these requirements and, moreover, offer the possibility of network-independent service creation.
The design principle of a softswitch with common generiac call control enables the network provider to offer his network capabilities to third party service providers in a suitable manner. In this architecture the creation of services is very easy and the traffic to the application is very low. Network providers' own gateways could be fitted in this architecture as well. The necessary changes, mainly in the call control layer, could be severe, and so this is an offensive architecture in terms of convergence.
The two proposed softswitch architectures with distributed call control differ only in the controllability of the gateway. If a network provider runs his own gateways, he will try to prefer the distributed call control instead of the pseudo softswitch. Both architectures have in common that here the legacy systems could mostly be reused. With the usage of Parlay, existing IN applications are utilizable, and, except for the integration of Parlay, no changes in call control are necessary. Support of other networks and extension of functionality could be added step by step in module manner. So the distributed is approach a moderate architecture in terms of convergence.

4 Location Based Services

This thesis has so far discussed the general requirements, designs and architectures of service platforms for next generation telecommunication networks. Telecommunication service platforms enable the establishment of a new class of service providers and independent software vendors that design innovative services accessing the network through open and standardized third party interfaces such as OSA/Parlay. Service oriented computing complying with a Service Oriented Architecture (SOA) has been identified as a significant principle of design that enables the deployment of services that scale to millions of subscribers.

Another key factor for these services to be successful is their personalization, i.e., the ability to take into account user's preferences, presence, or location. This chapter will discuss the design and architecture of Location Based Services (LBS). LBS provide added value mainly by using the physical position of mobile users. Location data may consist of plain geographical coordinates, access point cell IDs, civil location in form of postal addresses or more abstract definitions like ‘in the office’, ‘at home’.

The IP Multimedia Subsystem (IMS) [Camarillo2006] has been chosen as the target core network for these LBS. The IMS is an overlay network on top of the UMTS packet service (GPRS). It has been specified by the 3GPP in the last years and is currently in the rollout phase. The IMS makes heavy use of the Session Initiation Protocol (SIP) and its extensions and defines several service enablers such as presence, instant messaging, and push to talk. IMS however is not a service platform per se, but provides a uniform and standardized way of handling quality of service, charging, roaming and access to service enablers.

This chapter discusses several design challenges that turn up when integrating a service platform offering LBS with IMS. First the privacy and security consequences of publishing location service interfaces as platform services have to be considered. third party service providers that use location information to add value to a basic service are less trusted than the network operator. The personalization need on the other side has lead to increasing privacy concerns and served as a motivation for research on efficient methods to protect user data. An efficient scheme has been developed by the author and his colleagues from the ftw research project P2 (Austrian patent number A 363/2004 [Pailer Pat 04]) that enables localisation of users by third party application while ensuring their privacy.

Furthermore a terminal-based location enabler architecture for the IMS is proposed [Pailer 05], [Pailer 06], [Fabini], [Reichl 06a], [Reichl 06b] that blends the requirements of LBS with the privacy and security architecture of the IMS presence system.

Before going into design details of the proposed solution however, a general overview on the IMS core network, the 3GPP location services and standardized third party location interfaces is given.

4.1 IP Multimedia Subsystem – IMS

Many mobile and fixed network operators already have started to migrate their telecommunication networks towards an All-IP infrastructure where voice loses its dominancy and becomes just one among many services. The IP Multimedia Subsystem [IMS], [Camarillo2006], [Cuevas], standardized by the 3GPP, is the most promising candidate for replacing legacy, voice-dedicated mobile networks with an All-IP technology. As opposed to traditional IP-based networks, the IMS guarantees end-to-end Quality of Service (QoS) in the network. The IMS creates an infrastructure that enables the fast deployment of new IP-based services and flexible billing while still maintaining compatibility with existing applications.
The IMS aims to:

- Provide Internet style services in telecommunication networks
- Realize the Mobile Internet
- Specify a common platform for multimedia services
- Capitalize on value added services

3GPP specification TS 22.228, "Service requirements for the Internet Protocol (IP) multimedia core network subsystem" [3GPP TS 23.228] defines the service requirements from users' and operators' perspective for the support of IP multimedia applications:

- Establishment of IP Multimedia Sessions
- Support of end-to-end Quality of Service (QoS)
- Interworking with Internet and circuit-switched networks
- Support of roaming
- Access independence (e.g., GPRS, WLAN, ADSL, cable networks)
- Policy control for the network operator
- Support of multiple public identities per user that can be associated with multiple terminals
- Specification of a service execution framework without the need to standardize each new service

4.1.1 IMS Core Network

The Session Initiation Protocol [SIP] has been chosen by the 3GPP for IP multimedia session control. SIP is based on design principles of the most successful Internet protocols, SMTP (Simple Mail Transfer Protocol) [RFC 2821] and HTTP (Hyper Text Transfer Protocol) [HTTP 1.1].
The Diameter protocol [RFC 3588] is used for AAA (Authentication, Authorization and Accounting) in the IMS network Diameter is an evolution of the RADIUS protocol [RFC 2865] that is widely used by Internet Service Providers (ISP). Diameter specifies a base protocol that can be extended by *Diameter Applications*. IMS defines Diameter Applications for interaction with the HSS or for billing and accounting.
The COPS (Common Open Policy Service) protocol [RFC 2748] is used to transfer policy decisions between Policy Decision Points (PDP) and Policy Enforcement Points (PEP).
MEGACO (Media Gateway Control) protocol [RFC 3015], also known as ITU-T Recommendation H.248, is used for the control of media gateways.
RTP (Real Time Transport Protocol) and RTCP (RTP Control Protocol) [RTP] are used to transport real-time media like audio or video.

Figure 4-1 shows an overview of the general IMS architecture. It has to be pointed out that the 3GPP IMS standards do not specify nodes but functions with standardized interfaces. The specifications do not describe how IMS functions shall be deployed to physical nodes. Interfaces are referenced by a two or three digit letter code. A complete list of IMS interfaces can be found in 3GPP specification [3GPP TS 23.002].
The user's terminal, called User equipment (UE) in the specifications, is attached to some kind of access network (wired or wireless) that is connected to the packet network such as a GPRS network.
The P-CSCF (Proxy-Call/Session Control Function) is the first point of contact in the IMS network. The P-CSCF acts as an inbound and outbound SIP proxy that is all SIP traffic from and to the terminal is routed through this node. The P-CSCF has an IPSec (IP Security) [RFC 2401] security association with the user's terminal. The P-CSCF may also include a Policy Decision Function (PDF) that controls access to media resources and QoS parameters. P-CSCF and GGSN are always located in the same network (visited or home). Currently most GPRS networks are configured that roaming UE attach to the home GGSN, so that the P-CSCF will also be located in the home network. The GGSN in the home network has no negative effect on the SIP signalling, but also results in media being always routed through the home network which may lead to an undesirable packet delays.
The I-CSCF (Interrogating-Call/Session Control Function) is a SIP proxy located at the edge of the home domain. A DNS query for the next SIP hop of a domain [RFC 3263] will return the address of an I-CSCF. The I-CSCF uses the Diameter protocol to contact the HSS (Home Subscriber Server) or the SLF (Subscription Locator Function) in order to find the appropriate S-CSCF (Serving-CSCF) for that user.
The HSS (Home Subscriber Server) is the central repository for user data, like location information, security data, user profiles and the address of the S-CSCF (Serving-CSCF). The SLF (Subscriber Locator Function) is a load balancer that maps user URIs to HSS instances.
The S-CSCF (Serving Call/Session Control Function) is essentials a SIP server and a SIP registrar. SIP registration means that the S-CSCF binds a public user identity (a SIP URI) to the IP address of the terminal, where the user is logged in. All SIP signalling to and from an IMS terminal is routed through the S-CSCF. The S-CSCF may forward the SIP requests to one or more application servers. The S-CSCF is connected to the HSS via the Diameter based Cx interface. The HSS provides authentication vectors during registration, stores user profiles and filter criteria for application server invocation and stores the S-CSCF address during an initial transaction that is retrieved by the I-CSCF. The S-CSCF also keeps users from performing unauthorized operations or establishing unauthorized sessions.
The Application Server (AS) implements services in the IMS network. The AS may act as a SIP proxy, a SIP redirect server, a SIP User Agent (UA) or a SIP Back-to-Back user Agent (B2BUA). There a three types of application servers, the native SIP AS, the OSA-SCS (Open Service Access-Service Capability Server) offering OSA APIs [OSA] to third party applications and the IMS-SSF (IP Multimedia Service Switching Function) allowing CAMEL (Customized Applications for Mobile network Enhanced Logic) services to work on top of an IMS network. An AS located in the home domain may use the Diameter based Sh interface to retrieve user related data from the HSS. An AS may also be located in a third party's domain.

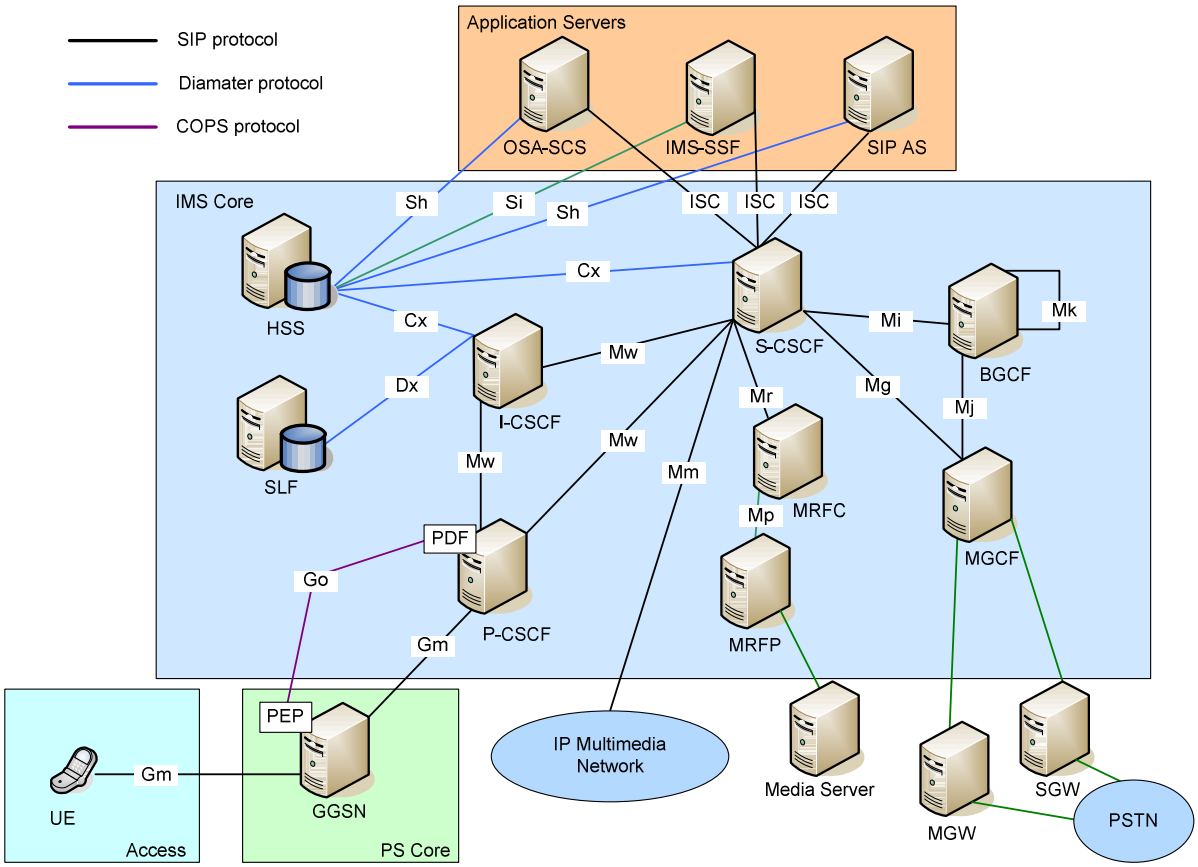


Figure 4-1: IMS overview

The Media Resource Function (MRF) is responsible for playing announcements, mixing media for a conference or transcoding media streams. The MRF is divided into a signalling plane MRFC (Media Resource Function Controller) and a media plane MRFP (Media Resource Function Processor). The MRFC is controlled by the S-CSCF using SIP. The MRFC controls the MRFP using the H.245/MEGACO protocol.

The BGCF /Breakout Gateway Control Function) provides SIP routing based on telephone numbers for calls from IMS terminals to and from circuit-switched networks. The MGCF (Media Gateway Control Function) implements a state machine that maps SIP to call control protocols of the circuit switched world. The SGW (Signalling Gateway) interfaces the signalling plane and the MGW (Media Gateway) the media plane for interconnection with a circuit-switched network.

The IMS-ALG (IMS-Application Layer Gateway) and the TrGW (Transition Gateway) (see Figure 4-2) are used for IP protocol conversion (IPv4 –Ipv6) and NAT (Network Address Translation) traversal. The IMS-ALG is a SIP B2BUA node that terminates a SIP call leg towards the terminal. The IMS-ALG exchanges the SDP (Session Description protocol) parameters (IP addresses and ports) of the terminal by IP address and port of the TrGW and establishes a new SIP call leg towards the final destination. The TrGW proxies RTP/RTCP packets between the address/port of the terminal and the address/port towards the destination. The IMS-ALG interfaces the I-CSCF for incoming and the S-CSCF for outgoing traffic.

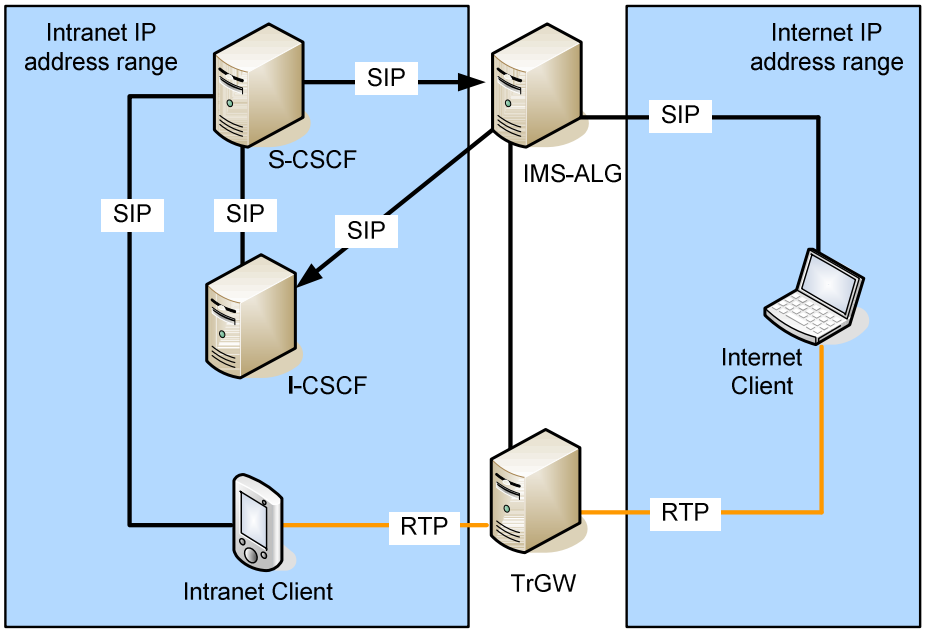


Figure 4-2: IMS-ALG for NAT traversal

4.1.2 IMS Identity Management

IMS introduces a novel user and service data model which defines the relationship between an IMS subscriber, its user identifier and the IMS service profiles. IMS subscribers are modelled as virtual entities with associated private user identifiers which identify a subscriber uniquely and are used for tasks like registration, authentication, authorization and accounting. Public identities following the SIP URI specification [SIP] or the TEL URL specification [RFC 2806] are linked to private identities and are utilized for requesting communication to other users. Private identifiers take the format of a Network Access Identifier [RFC 2486], e.g., *user@operator.com*. Compared to classic telephony networks, the private identifier is similar to the International Mobile Subscriber Identity (IMSI), whereas the public identifier corresponds to the Mobile Subscriber ISDN Number (MSISDN). Note that service profiles contain service specific information such as properties for time depending call-forwarding or presence settings which can be linked to one or more public identities of the subscriber.

IMS standard based authentication is done via the IMS Subscriber Identity Module (ISIM) [3GPP TS 31.103]. Similar to the Subscriber Identity Module (SIM), the ISIM is an application which resides on the Universal Integrated Circuit Card (UICC). It stores one private User Identity (UID) and one or more public UIDs which are allocated to the subscriber. While the private UID is used for authentication, the numerous public UIDs are used for the SIP compliant addressing of the subscriber. Figure 4-3 shows the correlation between the subscription and the several UIDs. Since IMS R6 it is even possible to use multiple private UIDs for the same subscriber which allows the subscriber to concurrently use multiple terminals with different UIDs. Furthermore the ISIM stores its home network domain name, as well as security parameters like the Cipher- and Integrity-Key.

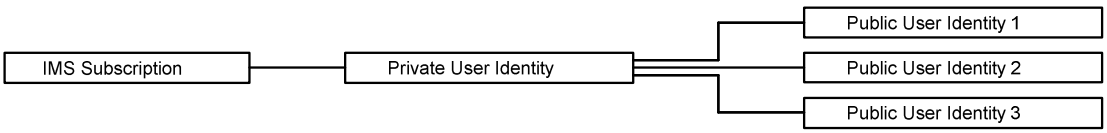


Figure 4-3: Relationship between Private and Public User Identities in IMS Release 5

Figure 4-4 illustrates the 3GPP Release 6 data model and shows a use-case where one subscriber owns multiple private identities (e.g., a private and a business SIM card) and concurrently uses several terminals. With the business SIM, he is reachable via his TEL-URI and also his business identity "user.business@operator.com". If the subscriber is registered only with his private identity, he is reachable via his TEL-URI and his private user identifier, while call forwarding settings apply for the TEL-URI and his presence status is announced only for his private identity.

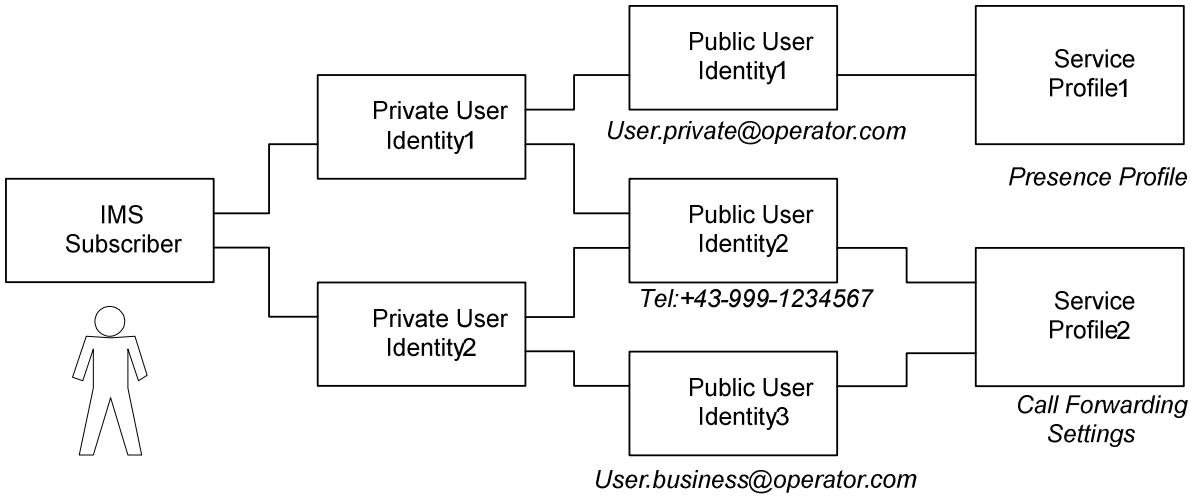


Figure 4-4: 3GPP IMS R6 user identity data model

In early IMS installations the ISIM application might not be available for the mobile terminals. Moreover also non-IMS subscribers should be able to authenticate and to use IMS services, allowing for smooth migration during IMS rollout. This can be achieved by using parameters stored in the 2.5G Universal Subscriber Identity Module (USIM) application for generating temporary IMS authentication credentials [3GPP TS 23.228]. The specified mechanism is to derive the required Private UID, Public UID and Network Domain URI from the International Mobile Subscriber Identity (IMSI) stored at the USIM. Thus, for the given IMSI in the mobilkom austria network "232011234567890" such a temporary version of an UID could for instance look like:
"sip:232011234567890@ims.mnc01.mcc232.3gppnetwork.org".
Another intermediate solution standardized by 3GPP re-uses information from the bearer level: if a user sets up a data call in a GPRS or UMTS network, he is already authenticated on the IP access level. [3GPP TS 31.103] specifies that the authentication server of the access network sends the identity from the IP boundary to the IMS core network, where this information is reused, thus transforming all users of GPRS/UMTS data services to potential IMS customers.
Resulting from the open and extensible data model and the circumstance regarding the standardized user entities, many issues and problems arise that have to be handled during the integration phase of IMS by operators:

- Most mobile operators do not have USIMs or ISIMs in the field up to now. Due to very high rollout expenses, intermediate solutions have to be utilized with limited functionality and security.

- Surrounding systems such as provisioning, billing, customer relations ship management engines have to be adopted to support the new data model and system capabilities which may result in high integration costs.
- Simple and easy to use products have to be defined, where the powerful data model capabilities and the new service capabilities are bundled combined.

4.1.3 IMS Security and Authentication

3GPP IMS provides a variety of measures to accomplish the high security requirements for the telecommunications domain. Most mechanisms use the "Authentication and Key Agreement" (AKA) mechanism executed between the customers USIM/ISIM and the HSS on the SIP level. AKA implements secure mutual authentication and keying material derivation on the core and user equipment side, the provided keying material is utilized for integrity protection and encryption. SIP control data is protected by an IPSec tunnel between the user equipment and the P-CSCF, where the AKA derived keys are utilized for IPSec tunnel keying. At the time being, the first operators deploy IMS production environments but there are many reasons why the security measures are not applicable as they were specified by 3GPP initially:

- USIMs and ISIMs are used only in a minority of deployed cellular networks, prohibiting the use of AKA based mechanisms. Moving from usual GSM SIM cards to USIMs/ISIMs would result in substantial costs for operators.
- Handsets do not support IPSec encryption.
- High bandwidth links like UMTS or HSDPA, over which encrypted SIP messages can be transmitted without harming the quality of service too much, are not widely available right now.

[3GPP TS 33.978] has defined some intermediate security measures to overcome the mentioned problems. The main idea is to re-use identity information from the bearer level instead of AKA for authentication on the SIP level. Session setups on the bearer level are terminated on the home GGSN, which sends RADIUS messages containing user MSISDN, IMSI and IP-address to an external authentication server. RADIUS accounting messages are forwarded to the HSS, which re-uses the provided information for extracting the user identity to IP relation for the IMS application level. When a client issues a REGISTER procedure, the identity is registered implicitly, without deriving keying material, therefore AKA dependent mechanisms are not used. Thus, an intermediate security scenario is available.

4.1.4 IMS Application Server

Services in the 3GPP IMS are realized by Application Servers (AS). 3GPP specification [3GPP TS 23.228] "IP Multimedia Subsystem (IMS)", describes the IP multimedia Subsystem Service Control Interface (ISC) as the interface between the Serving CSCF and the Application Server(s). ISC is realized by SIP but is named differently due to reasons.
The AS may either be located in the user's home network or in a third party domain. The AS hosts and executes value added services and may change session information on behalf of the services. More details about the interaction between AS and IP multimedia sessions can be found in 3GPP specification [3GPP TS 23.218] "IP Multimedia (IM) session handling". Transactions can be either forwarded to the AS by the S-CSCF or the AS can initiate SIP requests towards the S-CSCF.

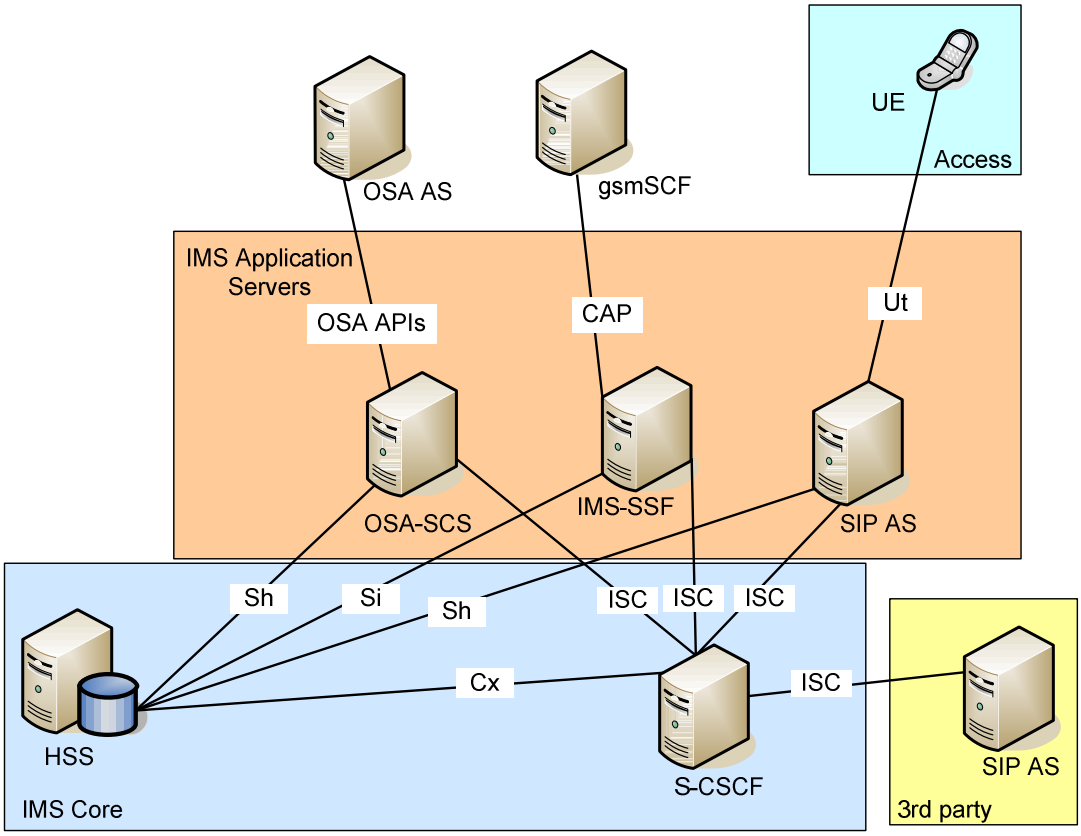


Figure 4-5: IMS Application Server

Figure 4-5 shows different IMS Application Servers (AS) and their interfaces. There a three types of application servers specified for the IMS, the native SIP AS, the OSA-SCS (Open Service Access-Service Capability Server) offering OSA/Parlay APIs [OSA] to third party applications and the IMS-SSF (IP Multimedia Service Switching Function) allowing CAMEL (Customized Applications for Mobile network Enhanced Logic) services to work on top of an IMS network. All types of AS use the ISC interface towards the S-CSCF.

The SIP AS is the most generic node to implement services in the IMS network. SIP AS that are located in the home network may query user data like user identities, location information, user status, initial filter criteria or charging information from the HSS using the Diameter based Sh interface (see [3GPP TS 29.328], [3GPP TS 29.329]).

The OSA-SCS application server is a gateway to third-party applications that use the OSA/Parlay APIs to invoke services of the core network. The OSA/Parlay APIs are network technology independent and can for example be used to implement converged services that combine services in a circuit-switched telephony network with services in an IMS infrastructure. From the viewpoint of the IMS core system, the OSA-SCS acts like any other SIP AS.

The IMS-SSF application server is a gateway to legacy service networks that implement CAMEL which is widely used in wireless networks. The IMS-SSF allows CAMEL services to invoked from the IMS core network. Form the S-CSCF view point, the IMS-SSF acts like any other SIP AS. The IMS-SSF can however access the HSS via the Si interface that is equivalent to the MAP (Mobile Application Part) protocol specified for GSM networks.

4.1.4.1 Application Server Session Model

SIP messages are forwarded by the S-CSCF to one or several SIP AS according to filter criteria that are part of the user profile and stored in the HSS. Filter criteria control, what services shall be executed.

A SIP AS may act as either an originating or terminating SIP User Agent (UA), a SIP proxy server, a SIP redirect server or as a SIP Back-to-Back User Agent (B2BUA). If a SIP AS decides not to provide a service, it should not record the route, so that further SIP messages are not forwarded to this AS once the current SIP transaction is completed.
The following message path diagrams show the basic SIP message flows between the S-CSCF and a SIP AS. The message flows show SIP INVITE messages as an example, but the flows are applicable to all initial SIP requests like INVITE, SUBSCRIBE, PUBLISH, OPTIONS. For the sake of clarity, the flows do not show provisional responses (like 100 Trying or 180 Ringing) between the initial SIP request and the final 200 OK.

4.1.4.2 Application Server acting as User Agent

Figure 4-6 shows the SIP AS acting as a terminating UA in the originating call leg, that is the AS becomes the destination of the call from the originating UA. An example for this use case is an announcement that informs the user, that he is not allowed to dial a particular number (call barring announcement).
The initial INVITE from the originating UA is proxied by the P-CSCF and received by the S-CSCF. The S-CSCF checks the initial filter criteria for the SIP URI in the INVITE message and decides to involve the AS. The AS acts as a terminating UA and responds with a 200 OK message that establishes the SIP dialogue.

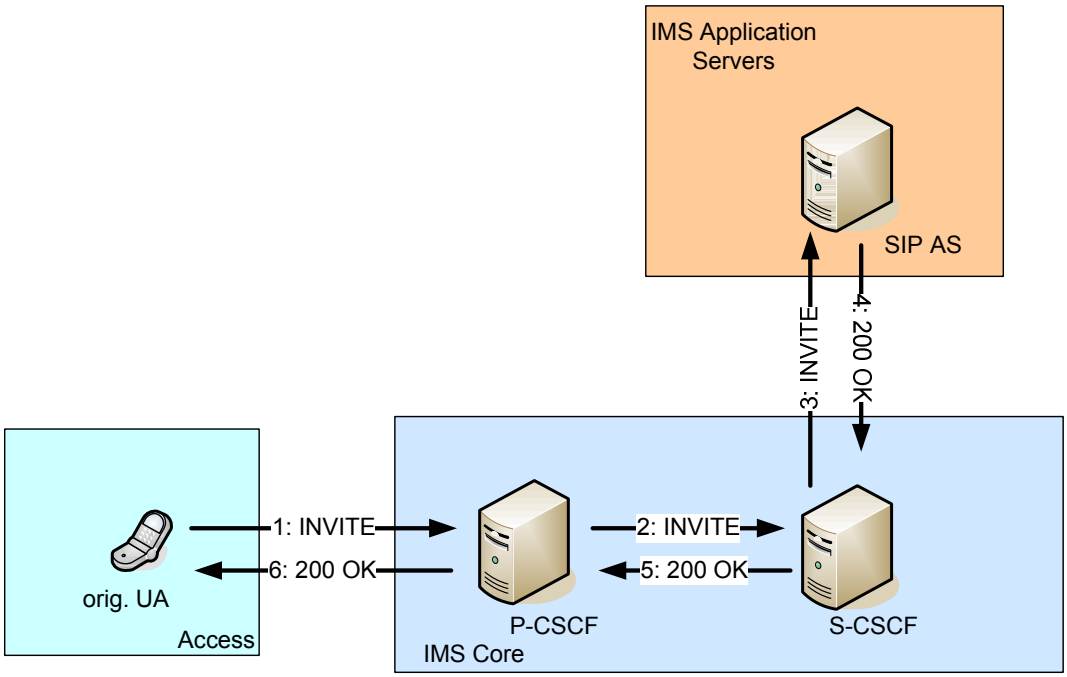


Figure 4-6: SIP AS acting as a terminating user agent in the originating call leg

Figure 4-7 shows the AS acting as a terminating UA in the terminating call leg. For the originating UA the AS becomes the destination of the call. The AS acts as UA on behalf of the original destination shown in the picture. An example for this use case is a voice mail application that is invoked unconditionally for the original destination.

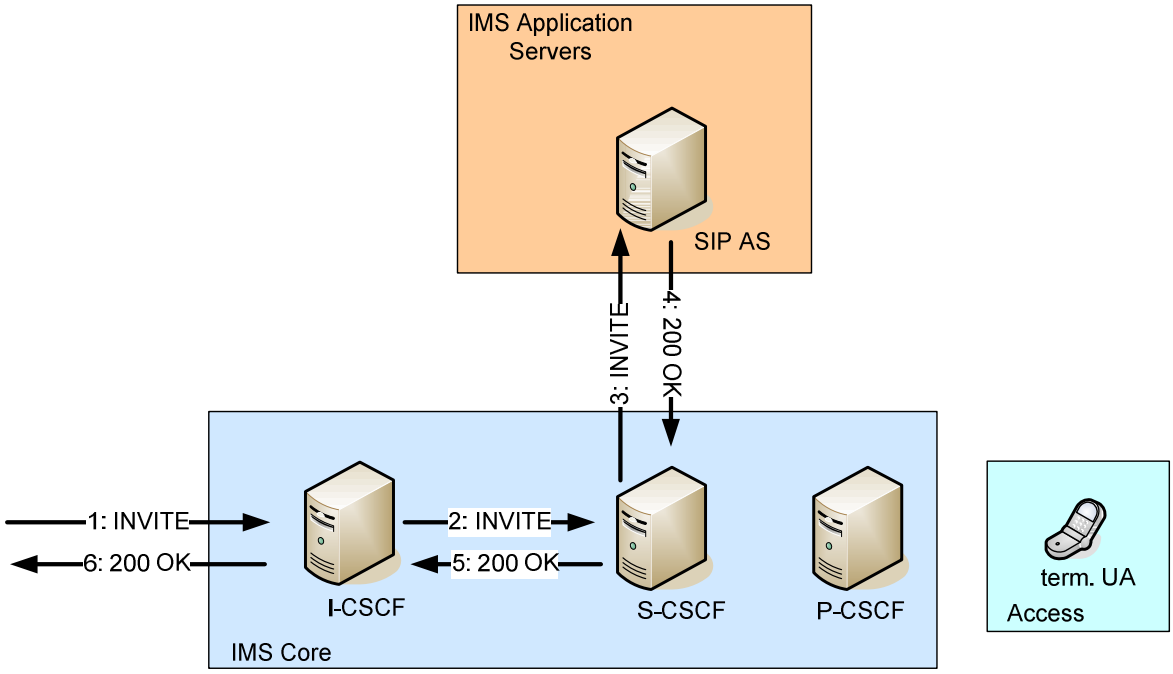


Figure 4-7: SIP AS acting as a terminating UA in the terminating call leg

Figure 4-8 shows the AS acting as an originating UA, that is the AS actually starts the call. An example for this use case is a wake-up call.

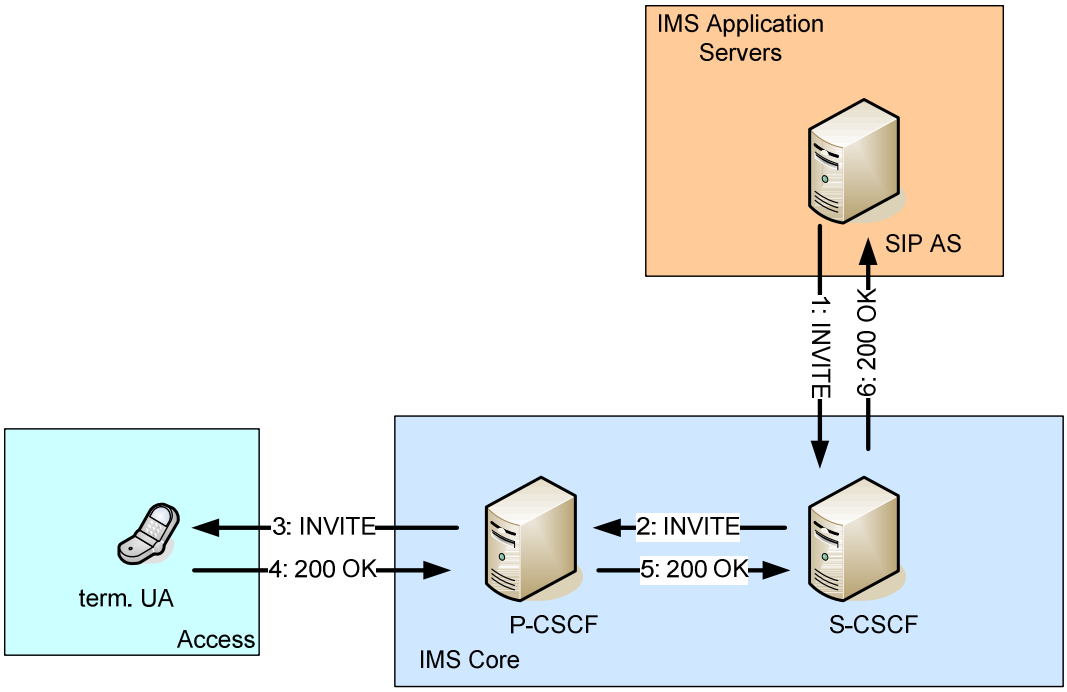


Figure 4-8: SIP AS acting as an originating UA

4.1.4.3 Application Server acting as a SIP Proxy

Figure 4-9 shows an AS that acts as a SIP proxy in the originating call leg. The INVITE message from the UA is proxied by the P-CSCF and then processed for the first time in the S-CSCF. The S-CSCF decides to forward the INVITE message to the AS. Before doing so, it adds a Route header to the message that contains the AS URI followed by the S-CSCF URI. The S-CSCF inserts some information to it's own URI that allows it to determine whether the message has already been sent to the AS.

The AS now provides its service, which may include manipulation of some SIP headers. A number translation service may e.g., alter the SIP URI of the INVITE request.
If the AS wants to be on the path of subsequent SIP transactions of that SIP session, the AS has to add its own AS SIP URI to the Record-Route header field. The S-CSCF will not forward the remaining SIP requests that belong to that SIP dialogue, to the AS, if the AS does not record its own route. The behaviour for SIP responses is different, as they will always take the same path (in reverse order) of the corresponding SIP request using the information in the Via headers.
When the AS has finished, the INVITE message is sent to the next URI in the Route header, which is the S-CSCF again. The S-CSCF gets the INVITE request for the second time and detects (e.g., from the username part of its own URI or from some URI parameter) that this message has already been processed by the AS and now forwards the INVITE request according to the SIP URI of the message.

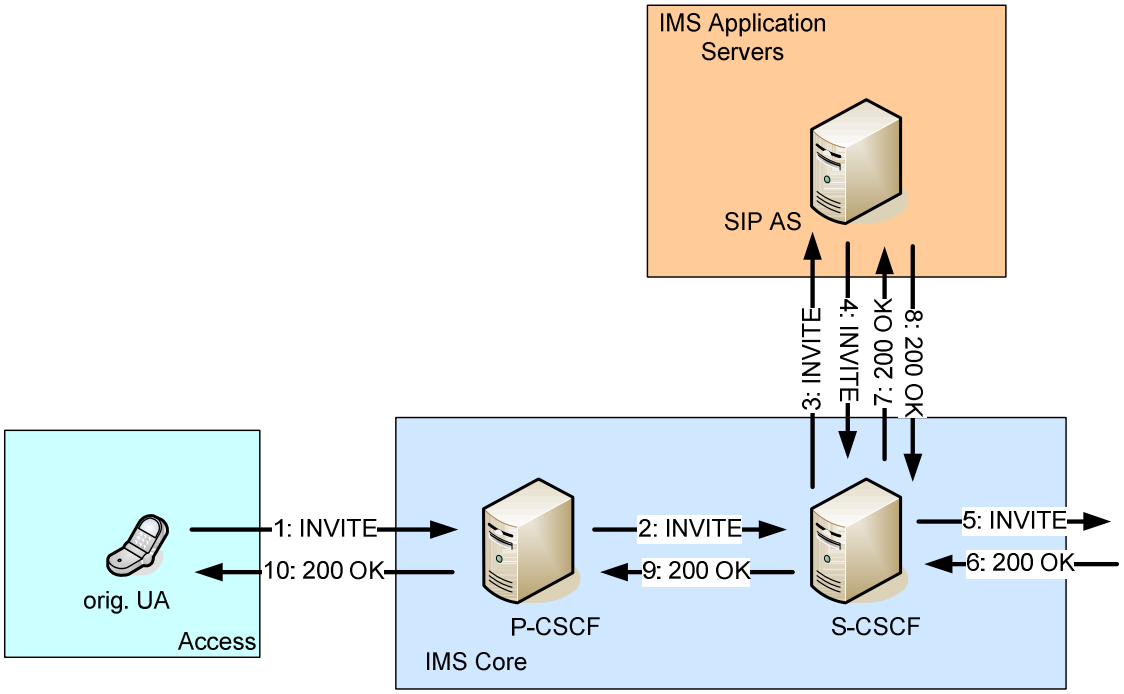


Figure 4-9: SIP AS acting as SIP proxy in the originating call leg

Figure 4-10 shows a SIP AS acting as a SIP proxy in the terminating call leg. Processing of SIP requests in the S-CSCF and the AS is similar to the scenario in the originating call leg. An example for this use case is a call forwarding service.

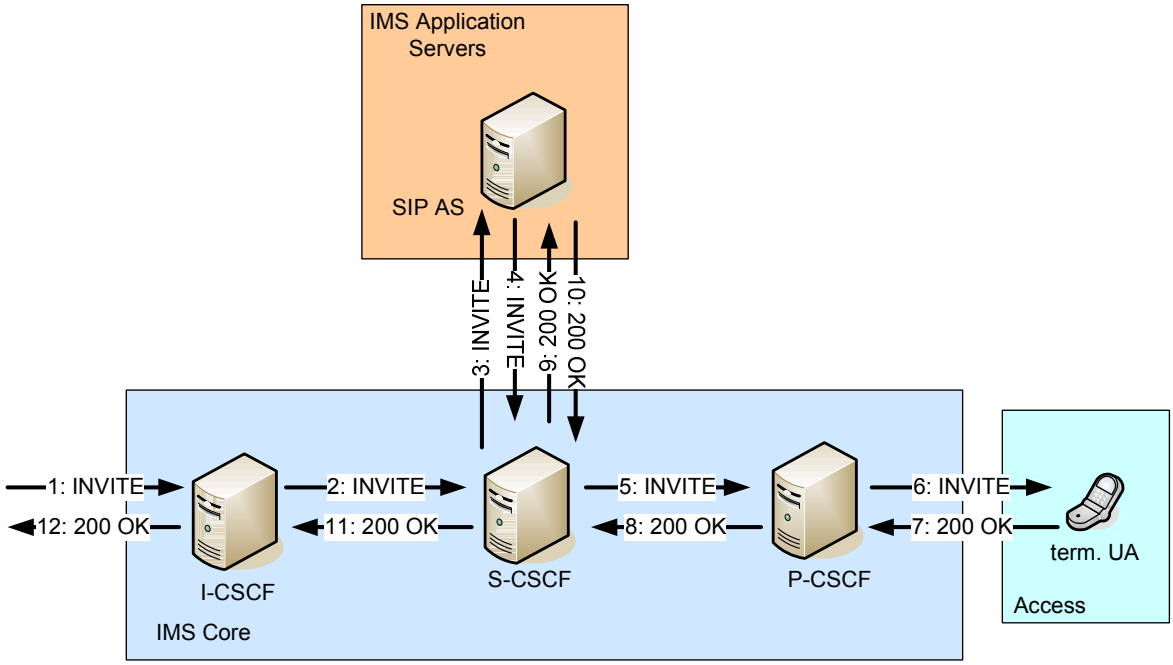


Figure 4-10: SIP AS acting as SIP proxy in the terminating call leg

4.1.4.4 Application Server acting as a SIP redirect Server

Figure 4-11 shows a SIP AS acting as a SIP redirect server in the terminating call leg. When getting an INVITE request, the AS generates a final 302 Moved Temporarily response and adds a Contact header that includes the new URI that shall be contacted.

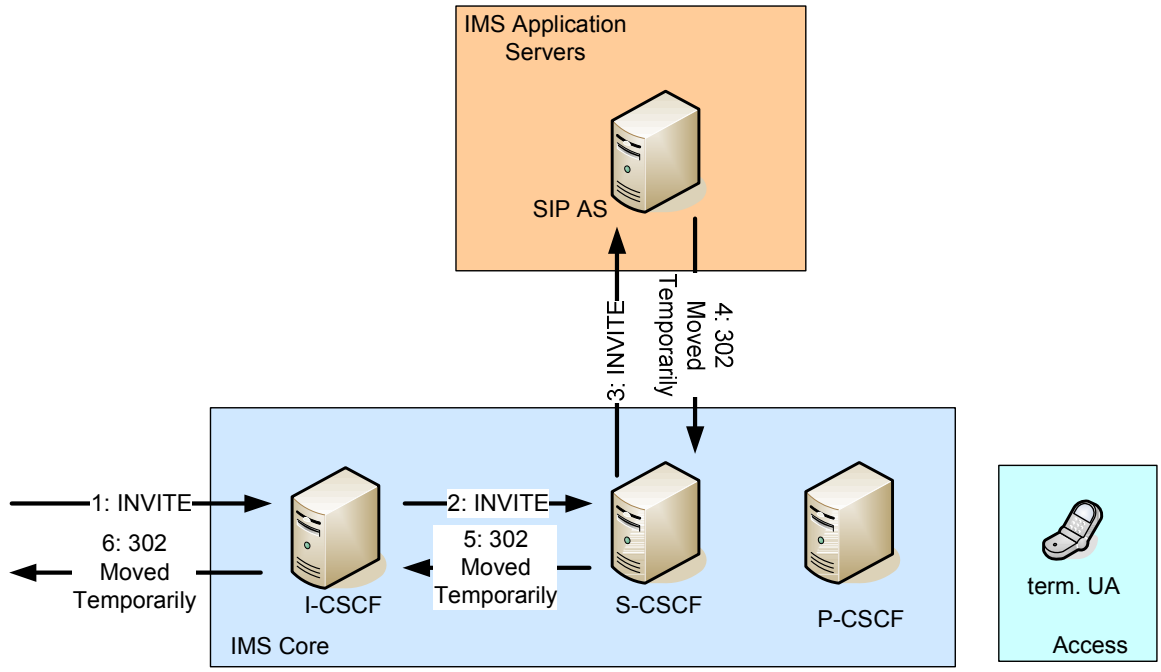


Figure 4-11: AS acting as SIP redirect server in the terminating call leg

4.1.4.5 Application Server acting as a Back-to-Back user Agent

Figure 4-12 shows a SIP AS acting as a B2BUA (Back-to-Back) in the originating call leg. A B2BUA acts as a UA server towards the A side of the call and as a UA client towards the B side of the call. Both call sides are independent SIP dialogues and only held together by the business logic of the B2BUA.

The difference between a B2BUA and SIP proxy is that a B2BUA is allowed to perform more actions than a SIP proxy. A B2BUA can manipulate headers like From, To, or Call-ID, it can manipulate the SDP body and it can send messages like a BYE. A B2BUA has to implement all functionality of a UA. This includes in the context of IMS also the SIP precondition extension (PRACK) for the support of QoS.
A B2BUA in the originating side could for example be used to implement a prepaid service. Once the account of the user is empty, the AS sends a BYE message to both sides of the call.

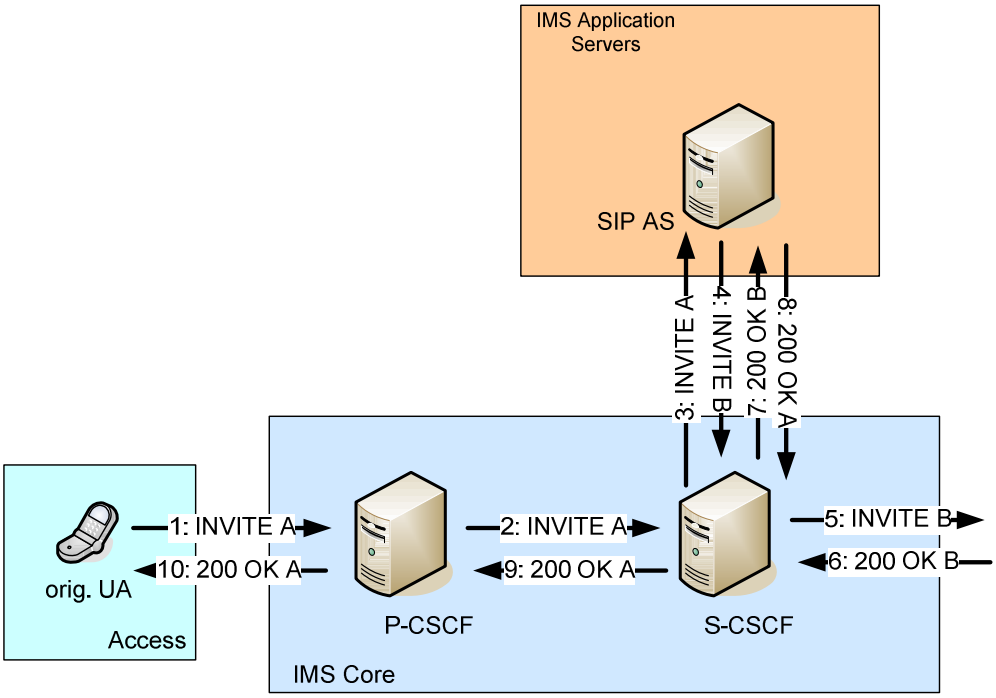


Figure 4-12: AS acting as B2BUA in the originating call leg

Figure 4-13 shows a SIP AS acting as a B2BUA in the terminating call leg. A B2BUA in the terminating call leg can be used to implement presentation restrictions of the originating user. The AS may replace the original From or Contact header by some anonymous token.

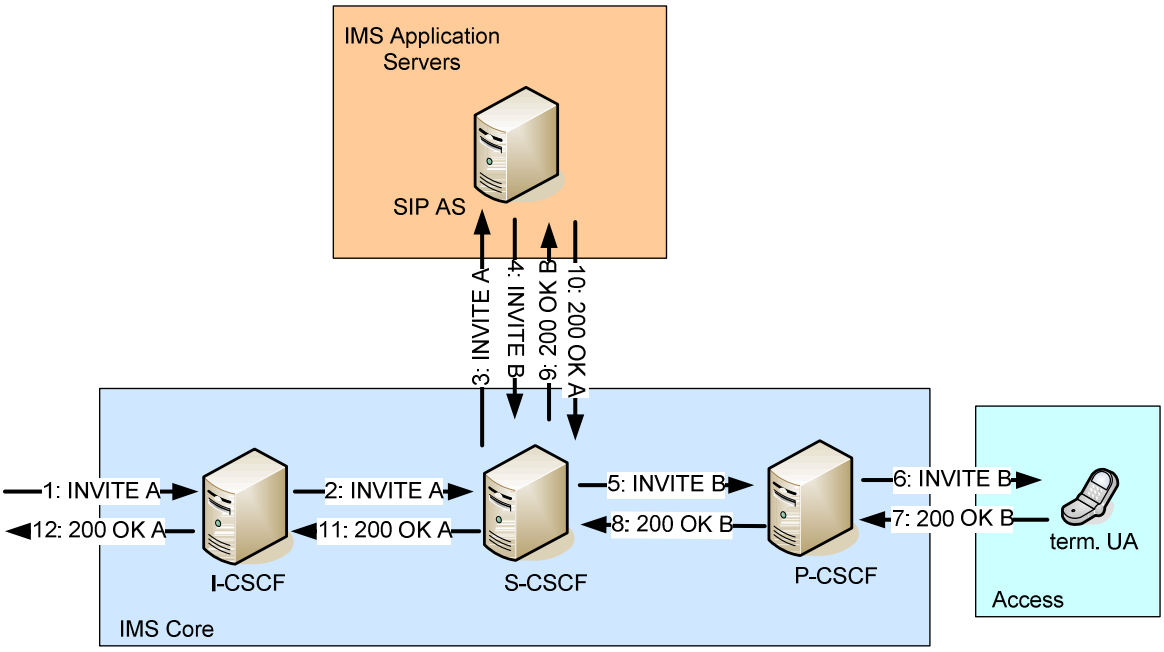


Figure 4-13: AS acting as a B2BUA in the terminating call leg

4.1.5 Filter Criteria

Filter criteria are stored in the HSS with the user profile and influence the decision of the S-CSCF what particular Application Server to invoke to provide a service. For historical reasons [3GPP TS 23.218] defines *initial* and *subsequent* filter criteria, but only initial filter criteria (iFC) are used. The S-CSCF checks the user's iFC when an initial SIP request (INVITE, SUBSCRIBE, OPTIONS, PUBLISH) is received that creates a new SIP dialog.

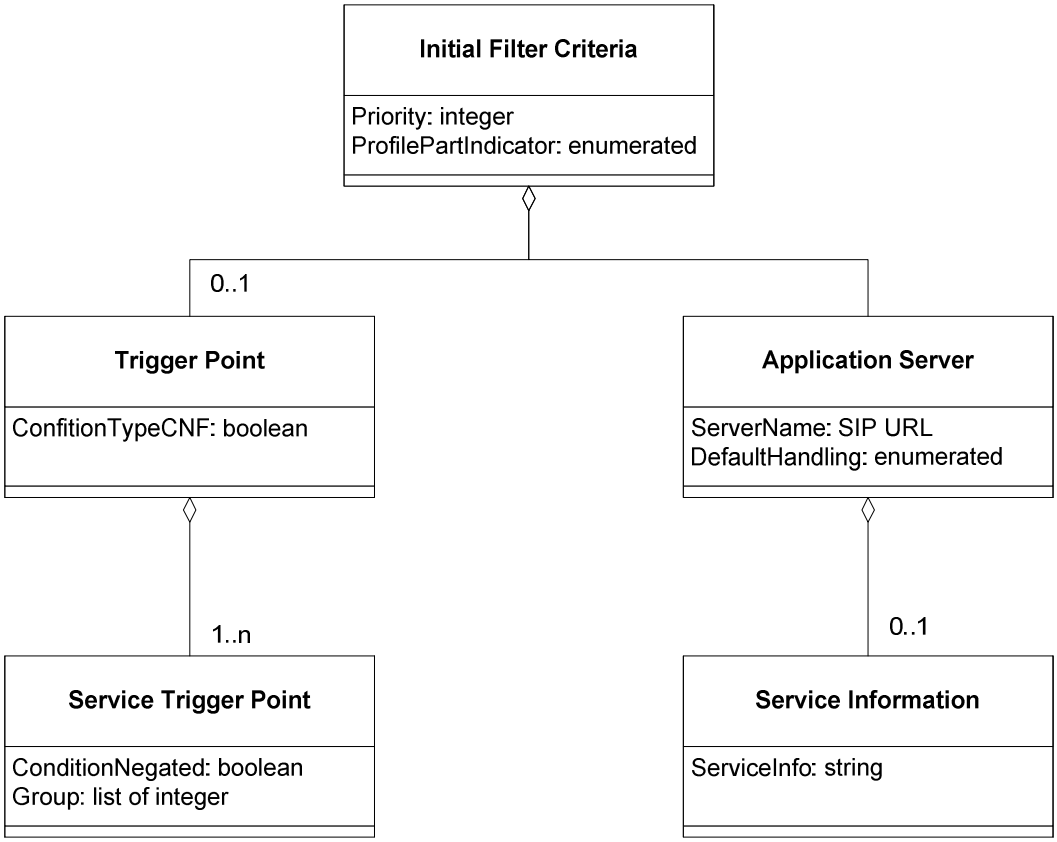


Figure 4-14: UML class diagram of the initial filter criteria

The user profile data structure in the HSS stores several service profiles for one private user identity. The service profiles can be attached to one or more public user identities. The iFCs are stored in the service profiles. Each iFC has a priority value assigned that has to be unique in the context of the service profile. A low priority value indicates high priority. iFC are processed in descending order of their priority value. This may result in several AS being contacted one after the other.

An iCF may define a Trigger Point that defines a rule that can be checked on the received SIP request, whether the iFC applies. Finally the iFC contains the AS URI where the received request should be forwarded to.

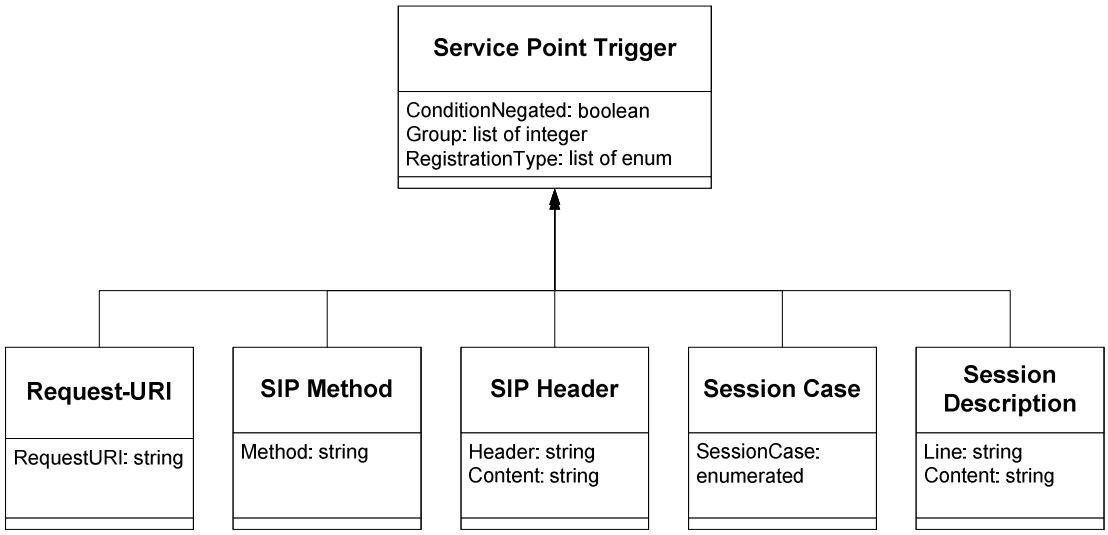


Figure 4-15: UML class diagram of the iFC Trigger Point

4.1.6 Application Server Access to the HSS

The Diameter [RFC 3588] based Sh interface is used by a SIP AS and an OSA-SCS to connect to the HSS. The Sh interface can be use to read and store user data in the HSS. Furthermore a subscription and notification mechanism allows an AS to get an event in the case of changes to the data stored in the HSS.

Implementation of the Sh interface is optional for the AS. The Sh interface, also know as the "Diameter Application for the Sh Interface" for the vendor 3GPP, is specified in [3GPP TS 29.328] and [3GPP TS 29.329].

User data in the context of the Sh interface includes the following:

- Public User Identifiers associated with the user
- S-CSCF allocated to the user
- IMS user state (registrered / not registered / pending)
- Initial filter criteria that apply to the requesting AS
- Charging Information (addresses of the charging functions)
- Repository data (transparent data for the service)
- User state (circuit-switched (CS) and packet-switcher (PS) user status)
- Location information for the CS or PS domain

Application Server Charging Architecture

The IMS charging architecture ([3GPP TS 32.200] for release 5 and [3GPP TS 32.240] for release 6) specify the Diameter protocol based Rf interface to send accounting information for offline charging from the Application Server to the CCF (Charging Collection Function) in release 5 and the CDF (Charging Data Function) in release 6. The AS receives the address of the CCF/CDF from the S-CSCF in the *P-Charging-Function-Address* SIP header. The CCF/CDF is then responsible to create CDRs that are sent to the billing system. Correlation of Rf messages are based on the *P-Charging-Vector* SIP header [RFC 3455] that is added by the P-CSCF to the SIP message flow.

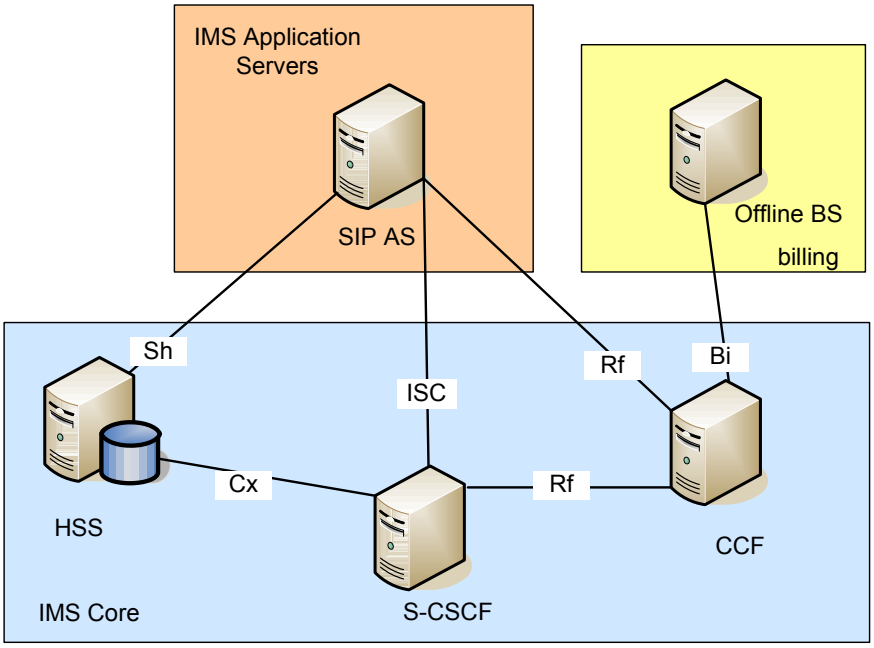


Figure 4-16: AS offline charging

For online charging the Ro interface is used between the AS and the Event Charging Function (ECF). The AS receives the address of the ECF from the S-CSCF in the *P-Charging-Function-Address* SIP header. In order to control online charged sessions the S-CSCF forwards all SIP messages to a special Application Server, the Session Charging Function (SCF). The SCF can terminate a session by sending BYE messages to all call parties, when the charged user runs out of credits.

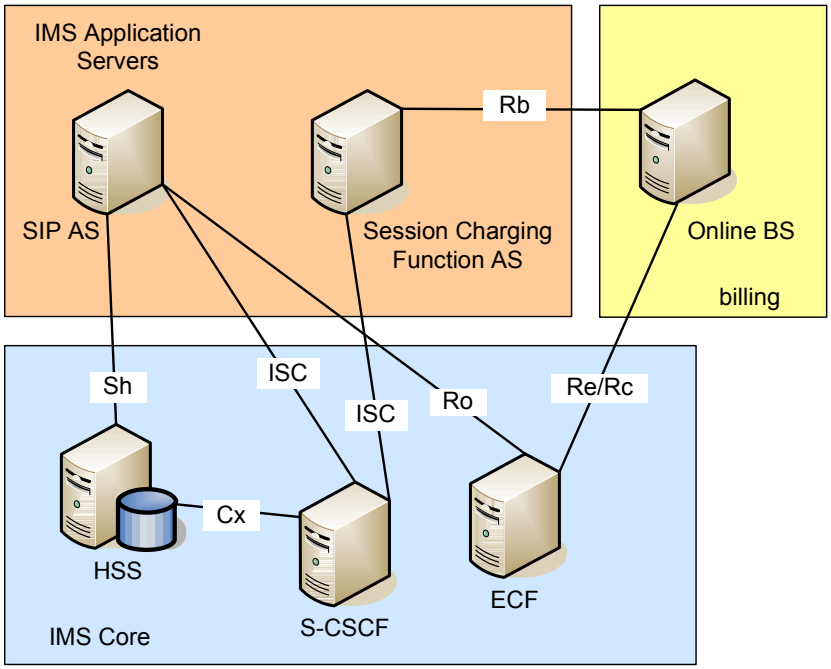


Figure 4-17: AS online charging

4.1.7 IMS Enabled Terminals

Success of services offered in an IMS network will largely depend on the availability of terminals that can be used to consume these services. Apart from a few prototypes, there are currently no mobile terminals on the market that feature IMS compatible user agent software. Mobile equipment manufacturers face some challenges when developing terminals that support IMS based multimedia services. The following list provides some of the most important user equipment requirements:

- Support of IMS style authentication (USIM/ISIM authentication)
- IPv6
- SIP protocol stack
- SDP protocol stack
- RTP protocol stack
- XCAP protocol stack
- XML parser
- Audio/video codec implementations (either in hardware or software)
- Encryption
- Concurrent execution of several sessions/connections (e.g., voice, video, instant messaging, presence)
- Integration of IMS call history, message history, contacts, buddy lists or presence data with standard mobile phone applications like address book, caller list, or message store.
- Over the air provisioning of IMS configuration parameters

This list of requirements shows that only terminal hardware platforms with excellent CPU performance, large memory size, support for many active I/O ports and above all sufficient battery capacity will be usable to implement IMS services. Furthermore operating systems of the mobile terminals have to support concurrency on a level similar to a standard PC operating system.

In this context, our research group at the FTW has gained some practical experience by implementing a prototype IMS messaging and presence agent based on the Java 2 Platform Micro Edition (J2ME). By deploying this software on currently available phones the following lessons have been learned:

- Support for JSR-180 (SIP-stack) is still weak. A SIP stack that is available in source code (sip-for-me) had to be used in order to get presence and TCP support
- Available memory strongly restricts the number of java libraries that can be used (e.g., for XML processing).
- Processing delay of a large SIP message containing XML in the body can be larger that the retry timeout of the requestor.
- TCP over an UMTS network adds considerable delay to the communication (1.5 RTTs at connection set-up time and 1 RTT for every packet sent). 1 RTT of the UMTS network in our experiments was measured to be about 150 ms.

User experience regarding user interface, feature level, quality of service or costs of packet oriented multimedia services is currently built up in the Internet by closed services like Skype. The telecommunication industry is still facing a huge task to develop open standards based IMS services to a competitive level. IMS enabled user equipment plays a crucial role in this task because the user will judge the IMS system by the user agent software.

4.2 Location Based Services in Third Generation Networks

3GPP specification TS 22.071 [3GPP TS 22.071] gives a general description of location services (LCS) and service requirements for Third generation networks. By making use of the radio signals the capability to determine the (geographic) location of the user equipment (UE) or mobile station (MS) shall be provided. The location information may be requested by and reported to a client (application) associated with the UE, or by a client within or attached to the Core Network. Location Services may be considered as a network provided enabling technology consisting of standardized service capabilities which enable the provision of location based applications.

LCS is available to the following categories of LCS clients:

- Value Added Services LCS Clients – use LCS to support various value added services. These clients can include UE subscribers as well as non-subscribers to other services.
- PLMN Operator LCS Clients – use LCS to enhance or support certain O&M related tasks, supplementary services, IN related services and bearer services and teleservices.
- Emergency Services LCS Clients – use LCS to enhance support for emergency calls from subscribers.
- Lawful Intercept LCS Clients – use LCS to support various legally required or sanctioned services.

LCS is being developed by 3GPP in phases:

- 3GPP Release 99: LCS is supported in the circuit switched domain of the 3GPP core network (GMLC connected to MSC). UTRAN R99 specifications support cell coverage (i.e., cell identity) based LCS.
- 3GPP Release 4 (including both UTRAN and GERAN): LCS shall be supported in the circuit switched domain and in the packet switched domain including GPRS. LCS shall be supported in GERAN and in UTRAN FDD and UTRAN TDD. The positioning methods in UTRAN will be at least cell coverage based, IPDL-OTDOA (Idle Period Downlink- Observed Time Difference Of Arrival) and network assisted GPS. LCS support is to be included in the Open Service Architecture (OSA) including enhancements for the support of value added services, and support for the velocity parameter in the position request /response.

The accuracy that can be provided with various positioning technologies depends on a number of factors, many of which are dynamic in nature. As such the accuracy that will be realistically achievable in an operational system will vary due to such factors as the dynamically varying radio environments (considering signal attenuation and multipath propagation), network topography in terms of base station density and geography, and positioning equipment available. The accuracy for location services can be expressed in terms of a range of values that reflect the general accuracy level needed for the application. Different services require different levels of positioning accuracy. The range may vary from tens of meters (navigation services) to perhaps kilometres (fleet management). The majority of attractive value added location services are enabled when horizontal location accuracies of between 25 m and 200 m can be provided. Table 4-1 gives an overview of horizontal accuracy requirements for LCS applications.

· Location-independent	Most existing cellular services, Stock prices, sports reports
· PLMN or country	Services that are restricted to one country or one PLMN
· Regional (up to 200 km)	Weather reports, localized weather warnings, traffic information (pre-trip)
· District (up to 20 km)	Local news, traffic reports
· Up to 1 km	Vehicle asset management, targeted congestion avoidance advice
· 500 m to 1 km	Rural and suburban emergency services, manpower planning, information services (where are?)
· 100 m (67%) · 300 m (95%)	U.S. FCC mandate (99-245) for wireless emergency calls using network based positioning methods
· 75 m-125 m	Urban SOS, localized advertising, home zone pricing, network maintenance, network demand monitoring, asset tracking, information services (where is the nearest?)
· 50 m (67%) · 150 m (95%)	U.S. FCC mandate (99-245) for wireless emergency calls using handset based positioning methods
· 10 m-50 m	Asset Location, route guidance, navigation

Table 4-1 LCS application accuracy requirements [3GPP 22.071]

Figure 4-18 shows the logical reference model for LCS whereby an LCS Client is enabled to request location information for one or more certain target UEs from the LCS Server supported by a PLMN. The LCS Server employs a positioning function to obtain the location information and furnish the information to the LCS Client. The particular requirements and characteristics of an LCS Client are made known to the LCS Server by its LCS Client Subscription Profile. The particular LCS-related restrictions associated with each Target UE are detailed in the Target UE Subscription Profile. The LCS feature shall allow a Target UE to be positioned within a specified Quality of Service.

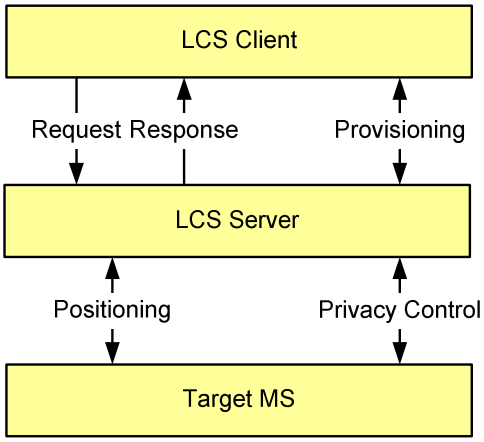


Figure 4-18: 3GPP LCS Logical Reference Model

Using the Location Service Request, an LCS client communicates with the LCS server to request the location information for one or more target UEs within a specified set of quality of service parameters. A Location Service Request is called ‘Immediate’ when there is only a

single response expected and 'Deferred' when one or more event driven response messages are possible.

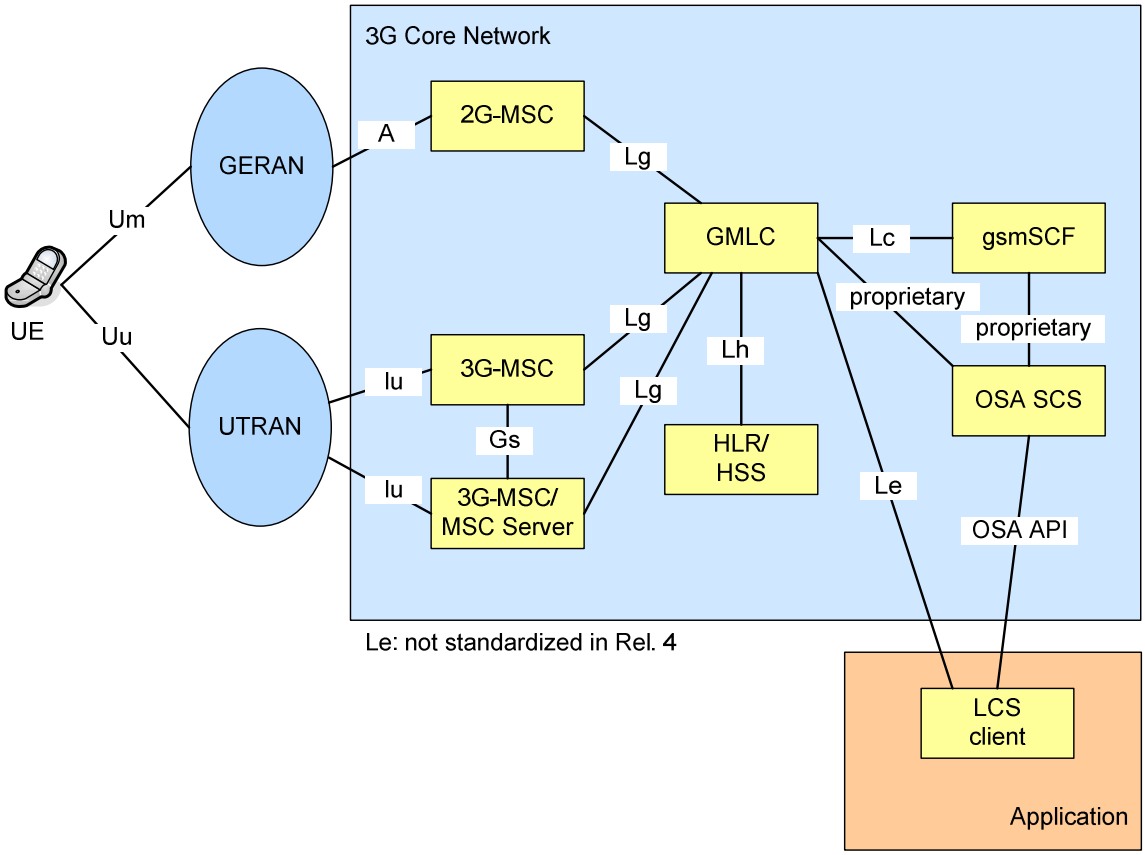


Figure 4-19 General LCS architecture specified by 3GPP [3GPP 22.071]

Figure 4-19 shows the general arrangement of the Location Service feature in GSM and UMTS according to the 3GPP specification TS 23.271 [3GPP TS 23.271] and illustrates the relation of LCS Clients and servers in the core network with the GERAN and UTRAN Access Networks.

The Gateway Mobile Location Centre (GMLC) contains functionality required to support LCS. In one PLMN, there may be more than one GMLC. The GMLC is the first node an external LCS client accesses in a GSM PLMN (i.e., the Le reference point is supported by the GMLC). The Lc interface supports CAMEL access to LCS and is applicable only in CAMEL. The Parlay/OSA Mobility SCF may also be supported by the GMLC. The GMLC may request routing information from the HLR via the Lh interface. After performing registration authorization, it sends positioning requests to either VMSC, SGSN or MSC Server and receives final location estimates from the corresponding entity via Lg interface. GMLC states during a location request are shown in Figure 4-20.

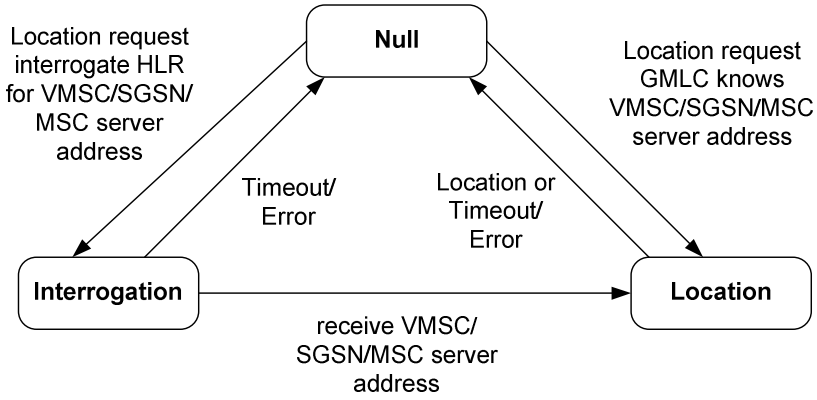


Figure 4-20 State Transitions in the GMLC

The target UE is identified and addressed by its MSISDN. It may be possible in certain cases to address the target UE using IP address when a static or dynamic IP address (IPv4 or IPv6) has been allocated for the UE. IP-addressing of the target UE is only possible when there is an active PDP context established between the target UE and the external LCS client.
The UE may be involved in the various positioning procedures. The UE interacts with the measurement co-ordination functions to transmit the needed signals for uplink based LCS measurements and to make measurements of downlink signals. The measurements to be made will be determined by the chosen location method. The UE may also, for example, contain an independent location function (e.g., Global Satellite Positioning Service GPS) and thus be able to report its location, independent of the RAN transmissions. The UE with an independent location function may also make use of information broadcast by the RAN that assists the function.
The MSC/VLR contains functionality responsible for UE subscription authorization and managing call-related and non-call related positioning requests of LCS. The MSC is accessible to the GMLC via the Lg interface. The LCS functions of MSC are related to charging and billing, LCS co-ordination, location request, authorization and operation of the LCS services. If connected to SGSN through the Gs interface, it checks whether the UE is GPRS attached to decide whether to page the UE on the A/Iu or Gs interface.
The HLR contains LCS subscription data and routing information. The HLR is accessible from the GMLC via the Lh interface. For a roaming UE, HLR may be in a different PLMN.

4.2.1 General Network Positioning Procedures

The generic network positioning procedure of providing the location information of an UE subscriber can be partitioned into the following procedures:

- Location Preparation Procedure
  This generic procedure is concerned with verifying the privacy restrictions of the UE subscriber, reserving network resources, communicating with the UE to be located and determining the positioning method to be used for locating the UE subscriber based on the requested QoS and the UE and network capabilities.
- Positioning Measurement Establishment Procedure
  This procedure is concerned with performing measurements by involving the necessary network and/or UE resources. Depending on the positioning method to be used for locating the UE the internals of this procedure can be positioning method dependent. The procedure is completed with the end of the positioning measurements.
- Location Calculation and Release Procedure

This generic procedure is initiated after the measurements are completed and is concerned with calculating the location of the UE and releasing all network and/or UE resources involved in the positioning

The general message sequence for a Mobile Terminal Location Request (MT-LR) is shown in Figure 4-21.

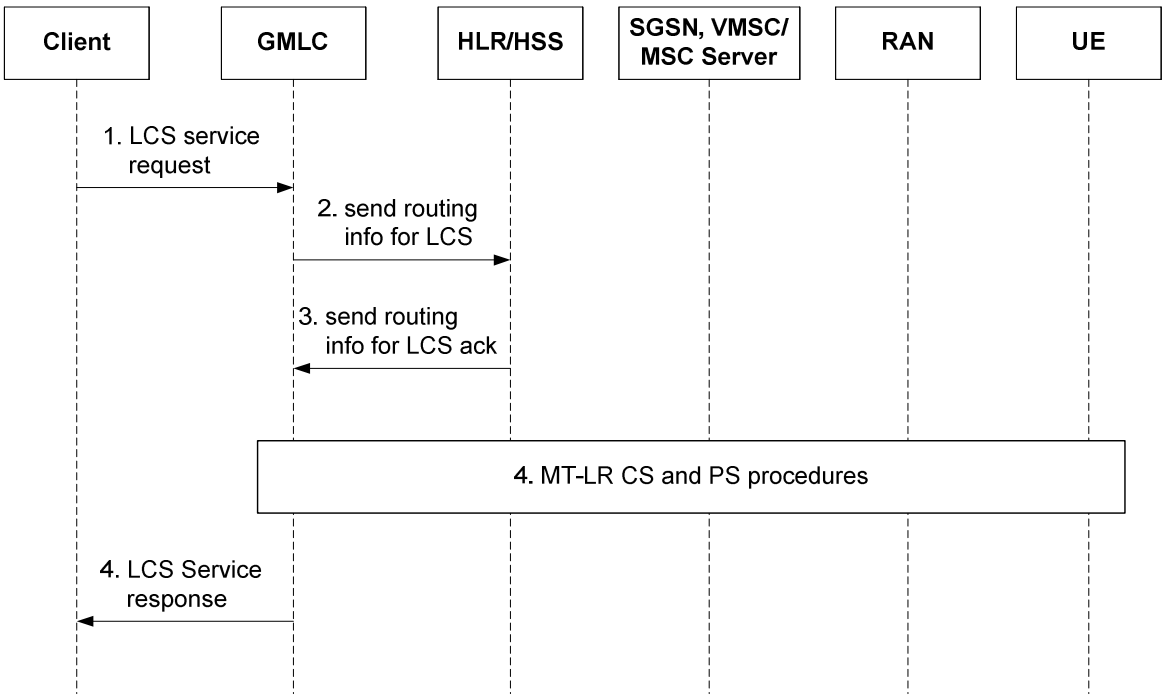


Figure 4-21 General Network Positioning for a Mobile Terminal Location Request (MT-LR)

1. An external LCS client requests the current location of a target UE from a GMLC. The GMLC verifies the identity of the LCS client and its subscription to the LCS service requested and derives the MSISDN or IMSI of the target UE to be located and the LCS QoS from either subscription data or data supplied by the LCS client. The GMLC obtains and authenticates the APN-NI of the LCS client. If location is required for more than one UE, or if periodic location is requested, the steps following below may be repeated.
2. The GMLC sends a SEND_ROUTING_INFO_FOR_LCS message to the home HLR of the target UE to be located with the IMSI or MSISDN of this UE. If the GMLC already knows both the VMSC/MSC server or SGSN location and IMSI for the particular MSISDN (e.g., from a previous location request), this step and step 3 may be skipped.
3. The HLR verifies that the GMLC is a known GSM/UMTS network element that is authorized to request UE location information. The HLR then returns one or several of the addresses, the current SGSN and/or VMSC/MSC server and whichever of the IMSI and MSISDN was not provided in step (2) for the particular UE.
4. GMLC proceeds with the Mobile Terminal Location Request procedure, either towards the SGSN (packet switched) or towards the VMSC/MSC (circuit switched).
5. GMLC sends the location service response to the LCS client. If the LCS client requires it, the GMLC may first transform the universal location co-ordinates provided by the SGSN or MSC/MSC server into some local geographic system. The GMLC

may record billing for both the LCS client and inter-network revenue charges from the SGSN or MSC/MSC server's network.

6.

Figure 4-22 illustrates a Packet Switched Mobile Terminating Location Request (PS MT-LR).

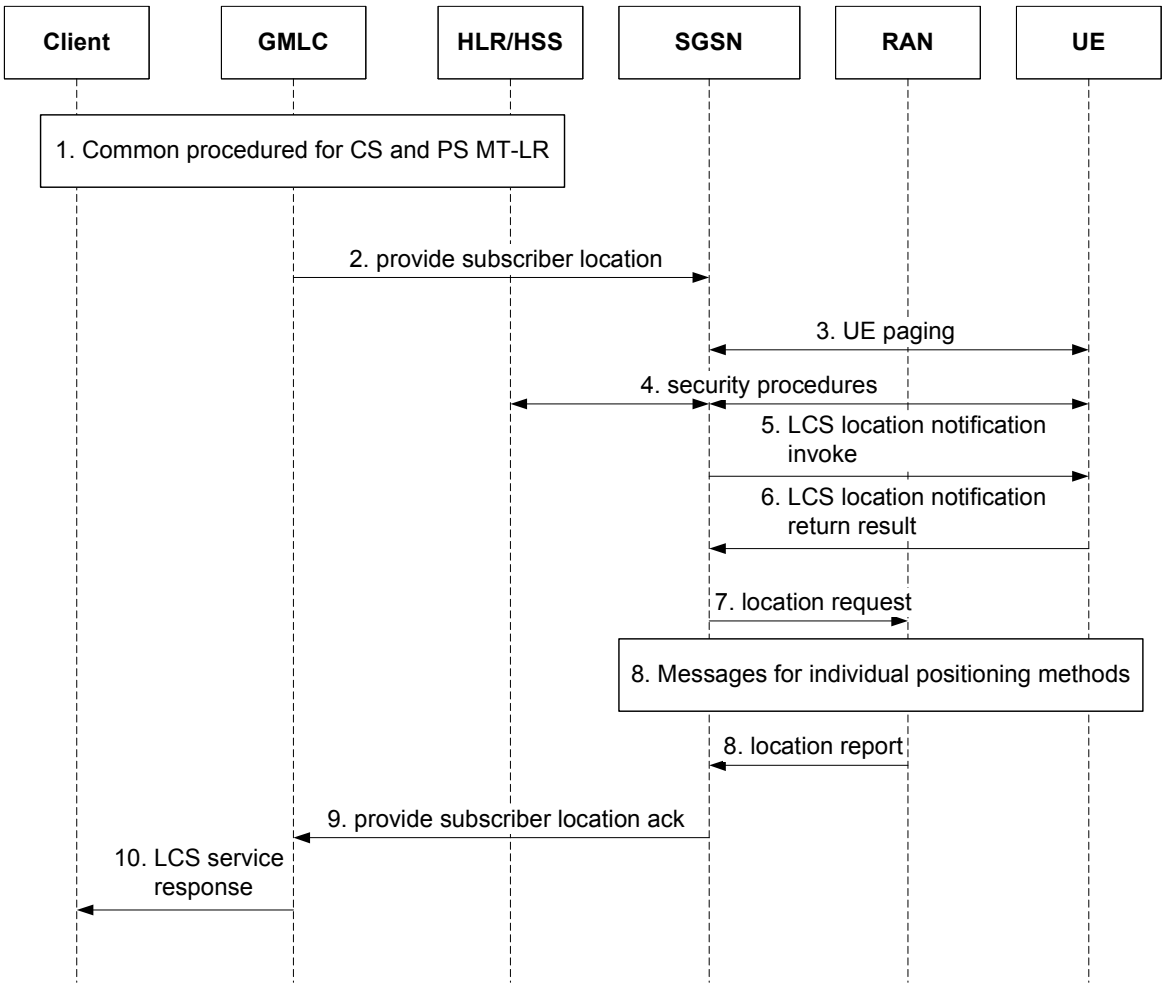


Figure 4-22 General Network Positioning for Packet Switched MT-LR

1. Common PS and CS MT-LR procedure as shown in Figure 4-21.
2. GMLC sends a Provide Subscriber Location message to the SGSN indicated by the HLR. This message carries the type of location information requested (e.g., current location), the UE subscriber's IMSI, LCS QoS information (e.g., accuracy, response time) and an indication of whether the LCS client has the override capability for privacy settings.
3. If the GMLC is located in another PLMN or another country, the SGSN first authenticates that a location request is allowed from this PLMN or from this country. The SGSN then verifies LCS barring restrictions in the UE user's subscription profile in the SGSN. If the GMLS is located in the same network and if the UE is in idle mode, the SGSN performs paging. The paging procedure is the same as in call set-up.
4. Security functions may be executed.
5. If the location request comes from a value added LCS client and the UE subscription profile indicates that the UE must either be notified or notified with privacy verification and the UE supports notification of LCS, a notification invoke message is sent to

the target UE indicating the type of location request (e.g., current location) and the identity of the LCS client and whether privacy verification is required. Optionally, the SGSN may after sending the LCS Location Notification Invoke message continue in parallel the location process, i.e., continue to step 7 without waiting for a LCS Location Notification Return Result message in step 6.

6. The SGSN notifies the target UE user of the location request and, if privacy verification was requested, waits for the user to grant or withhold permission. The UE then returns a notification result to the SGSN indicating, if privacy verification was requested, whether permission is granted or denied. Optionally, this message can be returned some time after step 5, but before step 10. If the UE user does not respond after a predetermined time period, the SGSN shall infer a "no response" condition. The SGSN shall return an error response to the GMLC if privacy verification was requested and either the UE user denies permission or there is no response with the UE subscription profile indicating barring of the location request.
7. The SGSN sends a Location Request message to the RAN. This message includes the type of location information requested, the requested QoS and any other location information received in paging response.
8. If the requested location information and the location accuracy within the QoS can be satisfied based on parameters received from the SGSN and the parameters obtained by the RAN e.g., cell information and timing information (i.e., RTT), the RAN may send a Location Report immediately. Otherwise, the RAN determines the positioning method and instigates the particular message sequence for this method. If the position method returns position measurements, the RAN uses them to compute a location estimate.
9. When location information best satisfying the requested location type and QoS has been obtained, the RAN returns it to the SGSN in a Location Report message. If a location estimate could not be obtained, the RAN returns a Location Report message containing a failure cause and no location estimate.
10. The SGSN returns the location information and its age to the GMLC. If the SGSN has initiated the Privacy Verification process in step 5, it returns the location information only, if it has received a LCS Location Notification Return Result indicating that permission is granted. The SGSN may record billing information.
11. The GMLC returns the UE location information to the requesting LCS client. If the LCS client requires it, the GMLC may first transform the universal location coordinates provided by the SGSN into some local geographic system. The GMLC may record billing for both the LCS client and inter-network revenue charges from the SGSN's network.

In case of a "Detached" or "Not Reachable" target UE, the last known location and a time stamp stored at the VLR, MSC Server or SGSN, may be returned to a LCS client requesting location information if the LCS client specifically requested the current or last known location.

4.2.2 Positioning Methods

Figure 4-23 shows the UTRAN UE Positioning Architecture. The SRNC (Serving Radio Network Controller), receives authenticated requests for UE positioning information from the CN across the Iu interface. RNCs manage the UTRAN resources (including Node Bs, LMUs, the SAS) the UE and calculation functions, to estimate the position of the UE and return the

result to the CN. SRNC may also make use of the UE Positioning function for internal purpose e.g., position based handover.

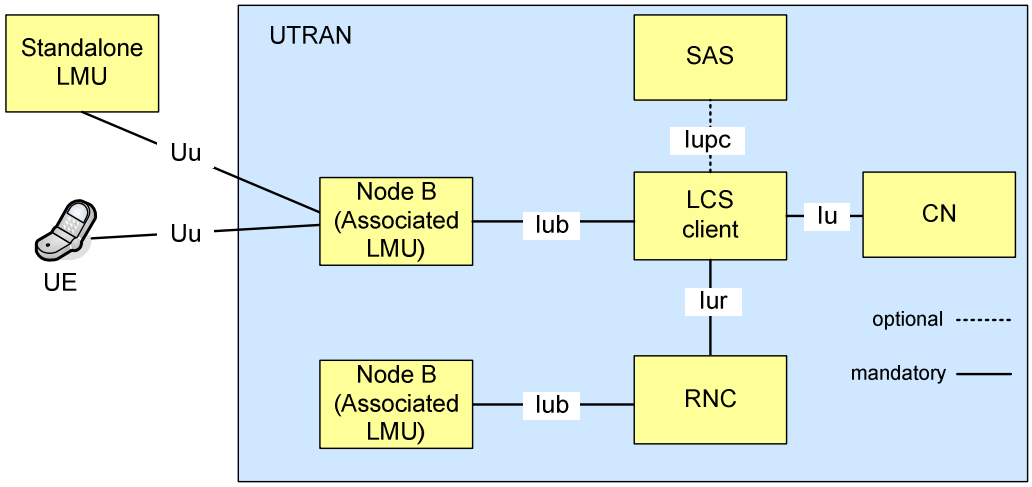


Figure 4-23 System components of UE Positioning in UTRAN

The standard positioning methods supported within UTRAN are:

- Cell ID based method
  In the cell ID based method, the position of an UE is estimated with the knowledge of its serving Node B. The information about the serving Node B and cell may be obtained by paging, locating area update, cell update, URA update, or routing area update. The cell coverage based positioning information can be indicated as the Cell Identity of the used cell, the Service Area Identity or as the geographical co-ordinates of a position related to the serving cell.
  The horizontal accuracy of a cell-id based localisation depends on the cell size, which is typically between 200 m-1000 m in urban areas and between 1 km-20 km in rural areas.
- OTDOA method
  The OTDOA-IPDL method involves measurements made by the UE and LMU of the UTRAN frame timing (e.g., SFN-SFN observed time difference). These measures are then sent to the SRNC and, in networks which include an SAS, may be forwarded to the SAS. Depending on the configuration of the network, the position of the UE is calculated in the SRNC or in the SAS.
  The horizontal accuracy of the OTDOA method is about 50 m – 150 m, but depends on the visibility of several Node Bs and thus has advantages in urban areas.
- Network-assisted GPS methods (A-GPS)
  These methods make use of UEs, which are equipped with radio receivers capable of receiving GPS signals.
  The horizontal accuracy of the network-assisted GPS method is 5 m – 10 m, but depends on the visibility of at least 4 GPS satellites. Satellite visibility if often degraded in urban areas.

Hybrid positioning methods have been proposed that combine radio network measurements like OTDOA and UE based GPS measurements in order to combine the strengths of both methods for rural and urban areas.

4.3 Third Party Access to Location Based Services

Data about a user's location is a typical type of information that may add considerable value to a service and that can be provided by the mobile network. A third party application needs access to network resources through proprietary or standardized APIs or protocols in order to get information about the location of a user. Location information may be provided as geographical coordinates or in form of cell IDs. This data can for example be used to show the user a map with the current location or to notify the user that the home/office cell has been entered or left (which may for example trigger a change of the user's profile). There are currently two major specifications that are standardizing access to location information. One is the "Mobility SCF" specification [Parlay Mob] by the Parlay Group that has been adopted by ETSI and based on top of it the Parlay X Web Service specification [Parlay X] that also contains WSDL definitions for getting location data. The second one is published by the former Location Interoperability Forum (LIF), that has consolidated as the OMA Location Working Group into the Open Mobile Alliance (OMA). The LIF has specified the Mobile Location Protocol [LIF], an XML based application layer protocol that defines an HTTP binding.

4.3.1 Parlay Mobility SCF APIs

The Parlay Group specifies a mobility API [Parlay Mob] that allows third party applications to locate users and to query user status by getting location data and status information from the network. The API is grouped in four services, the User Location service, the User Status service, the User Location Camel service and the User Location Emergency service.

The User Location service (UL) provides a general geographic location service. UL has functionality to allow applications to obtain the geographical location and the status of fixed, mobile and IP based telephony users. The UL service provides the `IpUserLocation` and `IpTriggeredUserLocation` interfaces. Most methods are asynchronous, in that they do not lock a thread into waiting whilst a transaction performs. To handle responses and reports, the clients must implement `IpAppUserLocation` and `IpAppTriggeredUserLocation` interfaces to provide the callback mechanism.

The User Status Service (US) provides a general user status service. US allow applications to obtain the status (reachable, not reachable, busy) of fixed, mobile and IP-based telephony users.

UL is supplemented by User Location Camel service (ULC) to provide information about network related information. Using the ULC functions, an application programmer can request the VLR Number, the location Area Identification and the Cell Global Identification and other mobile-telephony-specific location information.

There is also some specialised functionality to handle emergency calls in the User Location Emergency service (ULE). In the case of an emergency call, the network may locate the caller automatically. The resulting location is sent directly to an application that is dedicated to handle emergency user location using a callback method in the `IpAppUserLocationEmergency` interface.

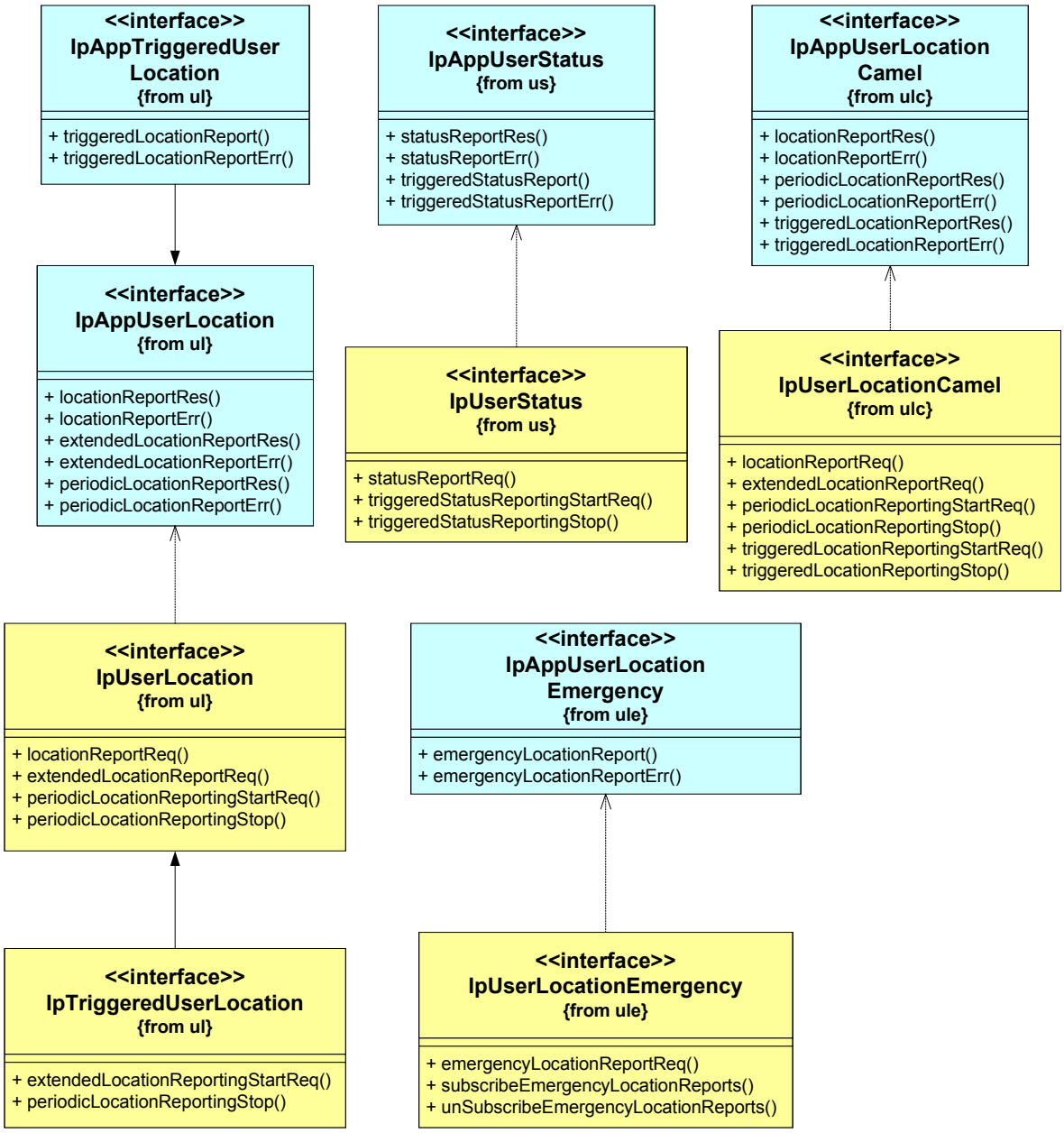


Figure 4-24 Parlay Mobility SCF Class Diagram

The class diagram for the Parlay Mobility SCF is shown in Figure 4-24. The message sequence chart in Figure 4-25 shows a triggered localisation. A start message is used to start triggered location reporting for one or several users. When the trigger condition is fulfilled then the network passes the location of the affected user to its callback object. The reporting is repeated until the application stops triggered location reporting.

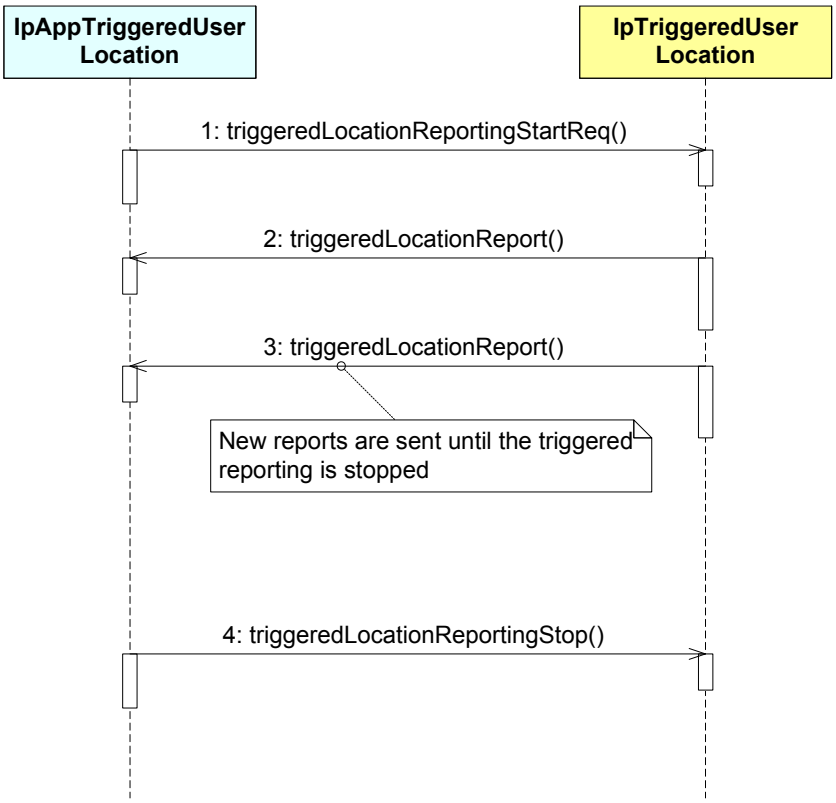


Figure 4-25 Parlay Mobility SCF Triggered Location Request Sequence

The Parlay Group has supplemented their API specifications with the Parlay X Web Service specification [Parlay X], that offers further abstractions and simplifications. It is the intention of Parlay X to define easy to use Web Services, that comply to the ideas of a service oriented architecture and that allow application developers that are not telecommunication experts to generate value added services based on telecommunication network capabilities. Parlay X provides WSDL specifications for the defined services.

The Parlay X Terminal Location Web Service can be used for getting location information:

```
getLocation(EndUserIdentifier endUser, EndUserIdentifier re-
quester, LocationAccuracy accuracy, out LocationInfo location)
```

The Web Service returns a location as longitude and latitude from the World Geodetic System 1984 (WGS 84).

The Parlay X User Status Web Service defines a single request that allows to query a user's status:

```
getUserStatus(EndUserIdentifier endUser, EndUserIdentifier re-
quester, out UserStatusData userStatus)
```

The status can be 'Online', 'Offline', 'Busy' or 'Other'.

When comparing the specifications for location services of the Parlay APIs and the Parlay X Web Services, one can immediately see, that the Web Service interfaces are reduced to a simple stateless location or status request transaction, whereas the API also offers asynchronous periodic or triggered location updates towards a callback application.

Most, if not all of today's services can be implemented with the simpler WSDL definitions from the Parlay X specification. The reason for this is that most of the currently deployed 2G, 2.5G and even 3G networks are not able to offer genuine triggered location updates, bat can only detect user status or user location changes by periodic polling. Polling however increases network load unnecessarily and does not scale to a large number of subscribers.

4.3.2 LIF Mobile Location Protocol

The Location Interoperability Forum specifies the Mobile Location Protocol MLP [LIF], that is an application-level protocol for getting the position of mobile terminals. The MLP serves as the interface between a Location Server (e.g., the GMLC in GSM and UMTS networks) and a location-based application. MLP specifies the following location services:

- Standard Location Immediate Service
  This service is used when a (single) location response is required immediately (within a set time) or the request may be served by several asynchronous location responses (until a predefined timeout limit is reached).
- Emergency Location Immediate Service
  This is a service used especially for querying of the location of a mobile subscriber that has initiated an emergency call.
- Standard Location Reporting Service
  This service sends the location data to a client, that has been set outside of the protocol.
- Emergency Location Reporting Service
  This is a service that is used when the wireless network automatically initiates the positioning at an emergency call. The position and related data is then sent to the emergency application from the location server. The application and its address are defined in the location server.
- Triggered Location Reporting Service
  This is a service used when the mobile subscriber's location should be reported at a specific time interval or on the occurrence of a specific event.

MLP is an application layer protocol that defines XML messages on top of a transport layer. Currently only HTTP is specified as transport protocol. An LCS Client requests a Location Service by issuing an HTTP POST request towards the Location Server. The message body of the request should include the XML formatted request. The response to the invocation of a Location Service is returned using an HTTP response. If a request is a deferred request (triggered or periodic) the result is delivered to the client through an HTTP POST operation issued by the Location Server. This implies that the client must be able to receive HTTP POST requests and be able to give a valid response.

4.4 Privacy in Location Based Services

Location and presence information will provide considerable value to information and communication services. Nevertheless, the users are still concerned about revealing their position data especially to un-trusted third party applications and legal restrictions regulate processing of personal data and the protection of privacy in electronic communications. In this section a novel privacy enhancement solution is proposed, which is targeted for location and presence services in the 3G service architecture and uses cryptographic techniques well suited to run in small devices with little computing and power resources. A patent [Pailer Pat 04] has been filed at the Austrian Patent Office for the proposed scheme. Once an observing user (called watcher) is granted the permission to localize another user (called presentity), the location server generates a key used to create pseudonyms that are specific for the localized user. Passed from the watcher to the location server via the application, these pseudonyms identify both the watcher and the desired localized user at the location server, but are opaque to the application. This section presents architecture and protocols of the proposed solution and discusses the performance increase in comparison with current implementations.

4.4.1 Motivation and Requirements

The possibility for a third party application to request a user's location from the network enables a whole group of value-added services that take into account the location of the user, either in form of geographical coordinates or reduced to presence information (e.g., 'in the office', 'at home'). The users however are concerned about privacy issues and hesitate to provide location information. Indeed, proposed standards and current implementations are full of security hazards that will grow with the appearance of services and additional clients for the network provided location information.

In the following some of the encountered problems are listed that motivated us to propose a solution that ensures the privacy of the located user:

1. In currently proposed standards (e.g., Parlay Mobility SCF [Parlay Mob]) and implementations, the LBS has to know the true identity of the target user, in most cases the MSISDN, in order to request location information from the network. Knowledge about the presentity's identity cannot be hidden completely, since the target has to be known to the network for completing the request for localization. This means that the presentity has to implicitly trust the network operator, but it is questionable if the presentity is willing to trust any third party that uses a location API provided by the network operator.
2. In most current cases, access control of the application to the presentity's location is handled at the LBS level. Thus, all relations of presentities to applications and to watchers requesting LBS information would have to be disclosed to the LBS.
3. It is the responsibility of the LBS to check the rights of watcher and application for the location of a certain presentity (also for non-repudiation and monitoring reasons). However, the watcher cannot be easily identified, e.g., he may access the application indirectly, anonymously or via the web.
4. As a consequence, the LBS need to configure access tables (watcher, presentity, application, operation, time)
5. The presentity cannot easily block the flow of location information to a certain watcher.

Furthermore the processing of personal data and the protection of privacy is regulated by law. The EU directive on privacy and electronic communications [EU Data Prov] mentions location data explicitly and establishes the following rules:

- Providers should minimize the processing of personal data and use anonymous or pseudonymous data where possible.
- Location data is listed as traffic data. Traffic data means any data processed for the purpose of the conveyance of a communication on an electronic communications network or for the billing thereof.
- Location data in the sense of traffic data means any data processed in an electronic communications network, indicating the geographic position of the terminal equipment of a user of a publicly available electronic communications service and may refer to the latitude, longitude and altitude of the user's terminal equipment, to the direction of travel, to the level of accuracy of the location information, to the identification of the network cell in which the terminal equipment is located at a certain point in time and to the time the location information was recorded.
- For the provision of value added services, the provider of a publicly available electronic communications service may process traffic data to the extent and for the duration necessary for such services, if the subscriber or user to whom the data relate has given his/her consent. Users or subscribers shall be given the possibility to withdraw their consent for the processing of traffic data at any time. The service provider must inform the users or subscribers, prior to obtaining their consent, of the type of location data other which will be processed, of the purposes and duration of the processing and whether the data will be transmitted to a third party for the purpose of providing the value added service.
- Where consent of the users or subscribers has been obtained for the processing of location data, the user or subscriber must continue to have the possibility, using a simple means and free of charge, of temporarily refusing the processing of such data for each connection to the network or for each transmission of a communication.
- Location data other than traffic data is regulated to the same extent as traffic data.
- Location data may only be processed when they are made anonymous, or with the consent of the users or subscribers to the extent and for the duration necessary for the provision of a value added service.

Privacy requirements are also influenced by the type of the localizing application. Localizing applications are categorized in 'Pull', 'Push' and 'Tracking', depending on the relationship between the localizing user, and the user to be localized:

- Pull-applications are always triggered by a user interaction where watcher and presentity are identical. An example is a user who requests the weather forecast for the current location via a WAP portal. Pull-applications can be regarded as noncritical, as the user always gives his content prior to the localization process. There is however a privacy issue, if the localization requires the disclosure of the user identity (e.g., in form of a telephone number).
- Push-applications are once subscribed by a user and then push location dependent content in form of unsolicited events. A user may e.g., subscribe to a localized weather forecast MMS service, that is sent out every morning. Again watcher and presentity are identical, but localization may occurs without an explicit user interaction. Push ap-

plications are more critical, as the user may be silently localized and a user profile may be compiled from a combination of location data and user identity.

- Tracking-applications allow one user to request the location of other users. Examples are fleet-management systems or so called 'friend-finders'. This is the most sensitive case, because watcher and presentity are not identical and the purpose of the application may reach from one-time localization to plain profiling. Tracking implies that the watcher has to know the identity of the presentity.

The listed application categories have in common, that the localization process is usually executed by some kind of middleware component, that may not be part of the trusted network provider's domain, but is operated by a third party service or content provider. In general there is thus a relationship triangle between watcher, application and presentity.

The identified privacy issues of LBS and legislative regulations lead us to a set of requirements:

- The presentity should be able to control access to his/her location information, and should be able to cancel or modify these access settings
- It should be possible to disable LBS by the presentity
- The presentity should impose a policy for providing his/her position
    - Always, in a certain situation (e.g emergency), never, on demand only, for certain duration
    - Context (time, mode, location)
    - To specified applications or user groups
- The presentity should control the accuracy of the provided location information
- The presentity could use session based or persistent ASID (anonymous subscriber ID) for identification
- The presentity should be able to create and use multiple identities

Privacy enhancement technologies (PET) address four basic ISO requirements [ISO 15408]:

- Anonymity
  Ensures that a user may use a resource or service without disclosing the user's identity. The requirements for Anonymity provide protection of the user identity. Anonymity is not intended to protect the subject identity.
- Pseudonymity
  Ensures that a user may use a resource or service without disclosing its user identity, but can still be accountable for that use.
- Unlinkability
  Ensures that a user may make multiple uses of resources or services without others being able to link these uses together.
- Unobservability
  Ensures that a user may use a resource or service without others, especially third parties, being able to observe that the resource or service is being used.

These privacy issues are subject of a number of research projects dealing with address privacy, location privacy, service access privacy or authentication privacy [RAPID IST].

Location privacy is a requirement that arises today in all 2G and 3G mobile networks that start to offer location based services. Terminal (and user) localization is either done by the user's terminal directly, equipped with a GPS receiver, or in most cases by the network, using the signal strength of a few neighboring cells listening to the terminal. The location information (expressed for example in geographical coordinates) can be used by the localized user

(push/pull applications) for example to direct a call to the nearest pharmacy or taxi, or by another user (tracking applications) who may run an application that schedules a service team. In either scenario there is an application or service that processes location data and that, in the worst privacy case, is owned by some third party commercial organization.
Any proposed schemes to improve privacy in mobile networks clearly have strong interoperability and standardization aspects, as they have to be approved by 3GPP or IETF. The IETF "Geographic Location/Privacy (geopriv)" Working Group has defined location privacy requirements [Geopriv Req] in which a Location Object (LO) plays the main role: it contains the location information to be transmitted from the Location Server entity (a part of the network operator) to the location requestor (watcher). The LO contains also the access rules for different users and is itself cryptographically protected. While this proposal is general and powerful, it implies that a large data amount has to be transmitted and requires from the watcher a lot of processing power.

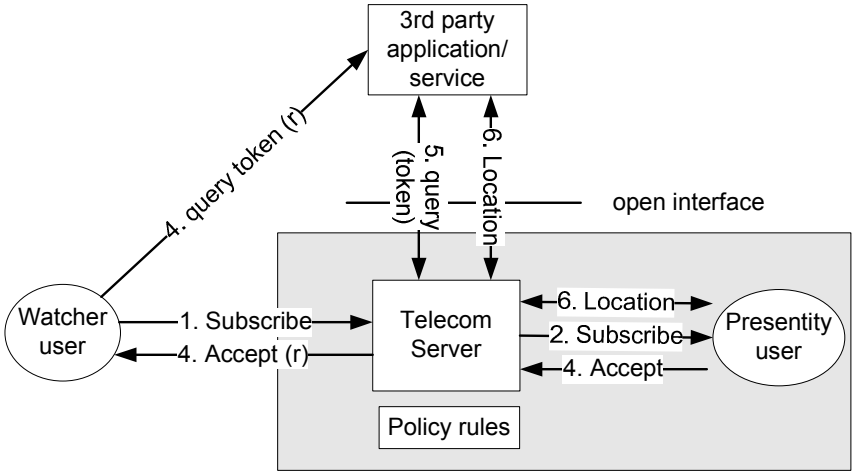


Figure 4-26 Architecture Overview

As the first GSM and UMTS networks start to offer location based services, the 3GPP has tried to improve privacy by tightening access control of users and applications on location information. Thus, in a privacy enhancement specification for UMTS Release 6 [3GPP LBS Priv], a complete authorization relationship between three entities: requestor (watcher), application and presentity has to be defined in the telecom server in order to allow access to location information. The responsibility of the Telecom Server in Figure 4-26 is to check the access rights of watcher and application (also for non-repudiation and monitoring reasons) and leads to complex processing, very large database tables and a tight coupling of all participating entities. The protection of the presentity identity is handled vaguely by proposing the use of aliases.
To overcome this situation, the approach described by Hauser et al.[Hauser 01] can be used in a modern service architecture. It is based on pseudonyms which are exchanged between the watcher and the telecom service in order to make it impossible for third party services to track the location of a certain presentity, store the location history or aggregate information from several services and create a profile. The difficulty in applying these schemes in the practice arises from the use of asymmetric encryption techniques and from the need of public key infrastructure, which implies creation of signatures and performing encryption as well as decryption processes which are computationally too expensive to be executed on today's mobile terminals.

These considerations have motivated our group to propose a more efficient scheme [Pailer Pat 04] to be used with the architecture in Figure 4-26 to protect the identity of localized users.

4.4.2 Service Interactions

This section describes the architecture and the generic interactions between watcher, presentity, Telecom Server and third party application. Figure 4-26 shows the service architecture with the following actors and components:

- **Watcher**
  is the user device that uses a third party application to requests the location of the Presentity within a dialog with that application.
- **Presentity**
  is the mobile user whose location is requested. It is able to respond to a subscription request from another user.
- **Telecom Server**
  authenticates the users that register their terminals and mediates requests from one user to subscribe to the presence/location info of another user as well as the decision of the requested user to grant or deny access to location info. The Telecom Server stores and forward the requests in case the requested user is offline.
  Furthermore it collects location information about users from the network.. The Telecom Server receives anonymous tokens form the third party application and retrieves the real identities of watcher and presentity and the authorized relationship between them.
  The Telecom Server is trused by the watcher and by the presentity.
- **Third Party Application**
  runs outside of the network provider's domain and is not necessarily trusted. The real identity of watcher presentity should not be revealed to the third party.

Basically, the watcher starts by establishing a trust relationship with the presentity using a subscription/notify message pattern. It is assumed that the presentity accepts a subscription to his/her location information only, if the watcher is known and trusted. Alternatively the acceptance rules may be predefined and stored in form of policies in the Telecom Server so that an interaction with the presentity is not required. Both approaches can be combined with a group management system to improve the usability of the system.
It has to be mentioned that the subscription/acceptance message exchange are preceded by the authentication of the users to the Telecom Server. Other messages are needed to query the status of subscriptions, which are stored and forwarded when the users go online:

- `getBuddies()` returns the list of subscribed and accepted (see below) presentities
- `getPendingWatchers()` returns the list of watchers waiting an accept message for that presentity
- `getPendingSubscriptions()` returns the list of presentities that did not send an accept yet.

Returning to the general operation in Figure 4-26, the accept message is mediated by the server which calculates a "root" r and sends it to the watcher (step 3). The root is used to compute a chain of one-time passwords (pseudonyms) that are subsequently sent in localization requests to the third party application. These pseudonyms are used to identify the presentity in the (standardized) API methods between the third party application and the Telecom Server (step 5). The pseudonyms sent by the application to the Telecom Server are used to

authenticate and authorize the localization request and to retrieve the real user identity. The Telecom Server then retrieves presence or location information of the presentity and returns this data to the application.

4.4.3 Use of Hash Values for Authentication and Authorization

A straightforward cryptographic solution to the mechanism above would require the use of digital certificates. Data has to be signed and/or encrypted in order to ensure authenticity and non-repudiation. Such algorithms have the disadvantage that they are computationally expensive and thus require high computing performance for signing or encrypting - an infeasible task for small devices such as mobile phones. In order to reduce the performance requirements on the watcher's and the presentity's device, a scheme is proposed, which authenticates and authorizes the watcher on the basis of one time passwords that are computed as elements of a hash value chains.

A hash value is the result of a hash function. A hash function H is a one-way function that it is easy to compute but computationally infeasible to invert. A hash function y = H(x) is defined as a function that for a given output y consisting on n bits the effort of computing x should require 2^{n-1} tries on average. Hash functions are also required to be collision resistant, that is finding two different inputs x and x' such that H(x) = H(x') requires an effort of $2^{n/2}$. The most commonly used hash functions are MD5 [MD5] and SHA-1 [SHA-1].

For repeated interactions such as requesting location information periodically, hash values are used that are calculated from the respective previous hash value.

One way hash-chains are a commonly used cryptographic technology. The idea to use hash values for authentication which are based on a chain of computations of hash values was first published by Lamport [Lamport]. Haller et al. [Haller] describe in the IETF RFC 2289 how to use hash chains for the computation of one-time passwords .

A hash chain is computed from a random value r that is called the root. This root value can for example be a combination of a secret only known to the user who wants to be authenticated and a random seed from the server that will authenticate the user. The root is used by the user to compute the hash chain. The length n of the chain is according to RFC 2289 determined by the user, but it is propose that the server sends n together with the seed. Value $p_n = H(r)$ is the n-th one-time-password (OTP) in the chain. The following OTPs are $p_{n-1}=H(p_n)$. The last value $p_0=H(p_1)$ generated is called anchor of the chain. The user sends the anchor p_0 back to the server where the value is stored. The indices of the one-time-passwords are given from a usage point of view. For the first authentication, the user sends p_1 to the server. The server authenticates the user by checking that $p_0 = H(p_1)$ and stores p_1 for the next authentication step. In general the server has to check that the hash of the last value received is equal to the last value stored, checking that $p_i = H(p_{i+1})$ (see Figure 4-27).

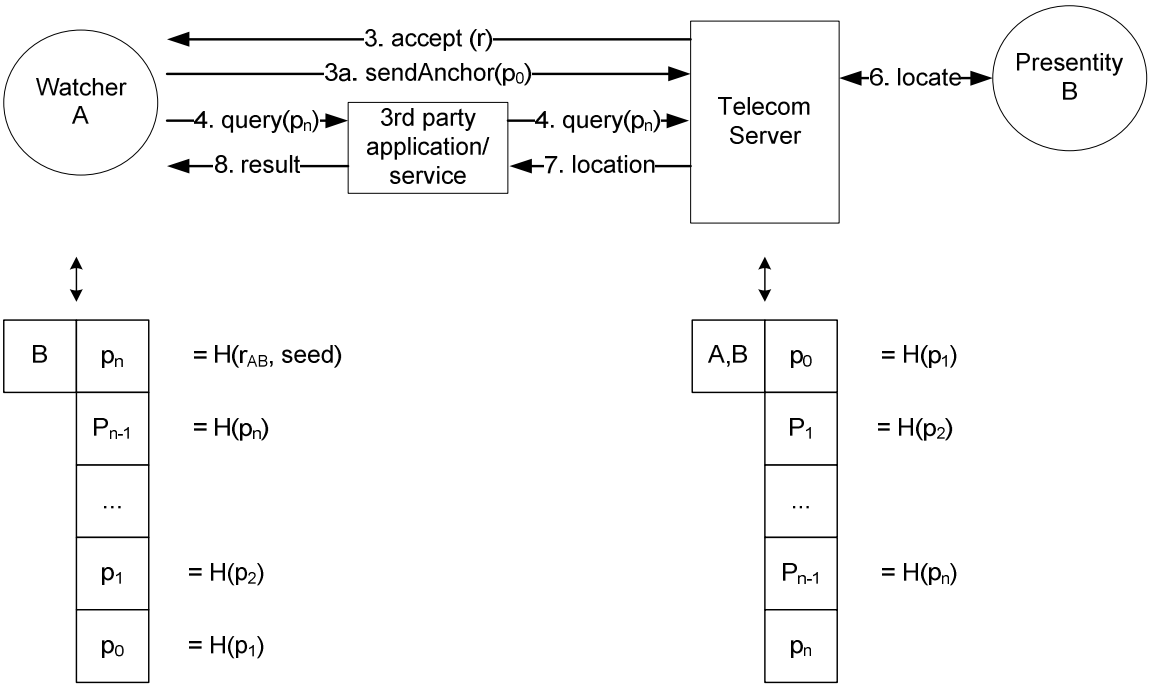


Figure 4-27 Hash Chain with reverse order one-time-passwords

The seed as well as the anchor and the one-time-passwords can be sent in clear, without an attacker being able to find out or forge the next password. This follows from the one-way property (difficult to invert) and collision resistance property (difficult to find another password that hashes to the last password) of hash functions.

There may however be the need to assure that the anchor is sent by an authorized user which means that previously to setting-up the hash chain the user has to be authenticated by the server.

The drawback of a hash chain based schemes is that the one-time-passwords have to be used in the reverse order of their creation, that is the i-th OPT p_i is the (n-i)-th value of the hash chain. This means that all OTPs could be computed at once and stored in the user's terminal, which is probably infeasible in mobile devices that usually have only a very restricted amount of memory available. Alternatively the OTP could be computed on demand from the root r, meaning that for every OTP the hash chain has to be partially rebuilt. On demand computation of the OTP p_i requires n-i applications of the hash function. It thus follows that for using all OTPs of a hash chain of length n, a mobile device would have to compute n*(n+1)/2 hash functions. The computational effort of calculating one OTP has the order O(n) and the on demand calculation of all OTPs of a hash chain of length n increases with the order $O(n^2)$ which also restricts the applicability on mobile device platforms.

The keyed hash scheme called HMAC [Bellare] overcomes the computational problems of the former schemes and still guarantees high security without the need of computation of a large number of hash values in advance. HMAC uses known hash schemes MD5 or SHA-1 and has a comparable efficiency. It allows us to create hash values in forward direction from a previous one, using a shared secret key, also called Message Authentication Code (MAC), to generate the hash values. The secret key has to be shared between the watcher and the telecom server and the HMAC procedure is performed on both sides (see Figure 4-28).

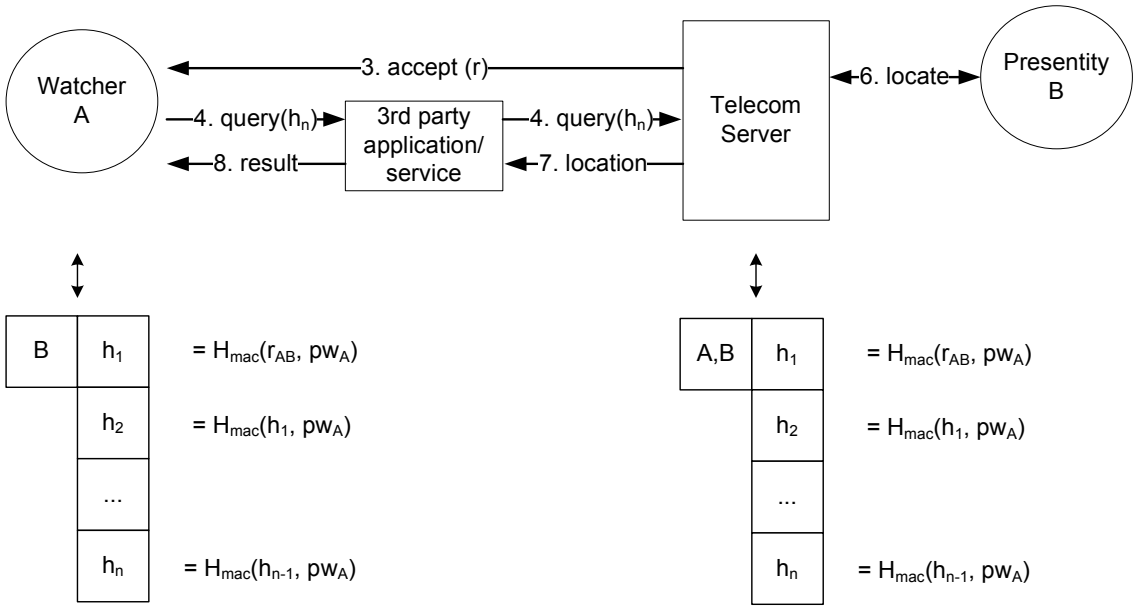


Figure 4-28 Hash Chain with forward order one-time-passwords using HMAC algorithm

The single information the watcher A needs to compute the hash chain is the root r_{AB}, which is initially created by the telecom server and is a function of the password, and a random number. The root r_{AB} is created if the subscription is successful (either if the policy rules grant access to that watcher, or the presentity sends an `acceptSubscription()` message). As shown in Figure 4-28, the hash values are created in parallel by the watcher and the telecom server form the previous value h_{n-1} and watcher's password by applying the HMAC operator $H_{MAC}()$.

Once the subscription is completed, the watcher can interact with the third party application using hash-values instead of presentity identifiers, typically calling `getLocation()` for example. The application passes the request to Telecom Server requesting `getLocation(h1)`. The Telecom Server receives the hash value `h1`, checks its validity and determines from the `h1` hash value the real identity of the watcher and the presentity. The message "location information" in Figure 4-29 refers to the localization procedure the network performs using cell information or other methods.

The correct operation of the system is based on the synchronization of hash values at watcher and server side. That means both sides calculate the next hash value to be used in the next `getLocation` request. However because of errors in the telecommunication channel or even malicious man-in the-middle attacks, messages can get lost, so that synchronization has to be re-established.

Another reason for the watcher and telecom server to get out of synchronization, is the unexpected crash of the client application. In case of the HMAC based hash chain scheme, which calculates hashes in forward direction, the current hash value has to be stored on the user's device after its use in a location request. For the next request, the stored value has to be read and together with the user's password the subsequent hash value is calculated. A fault while storing the current hash value breaks the chain.

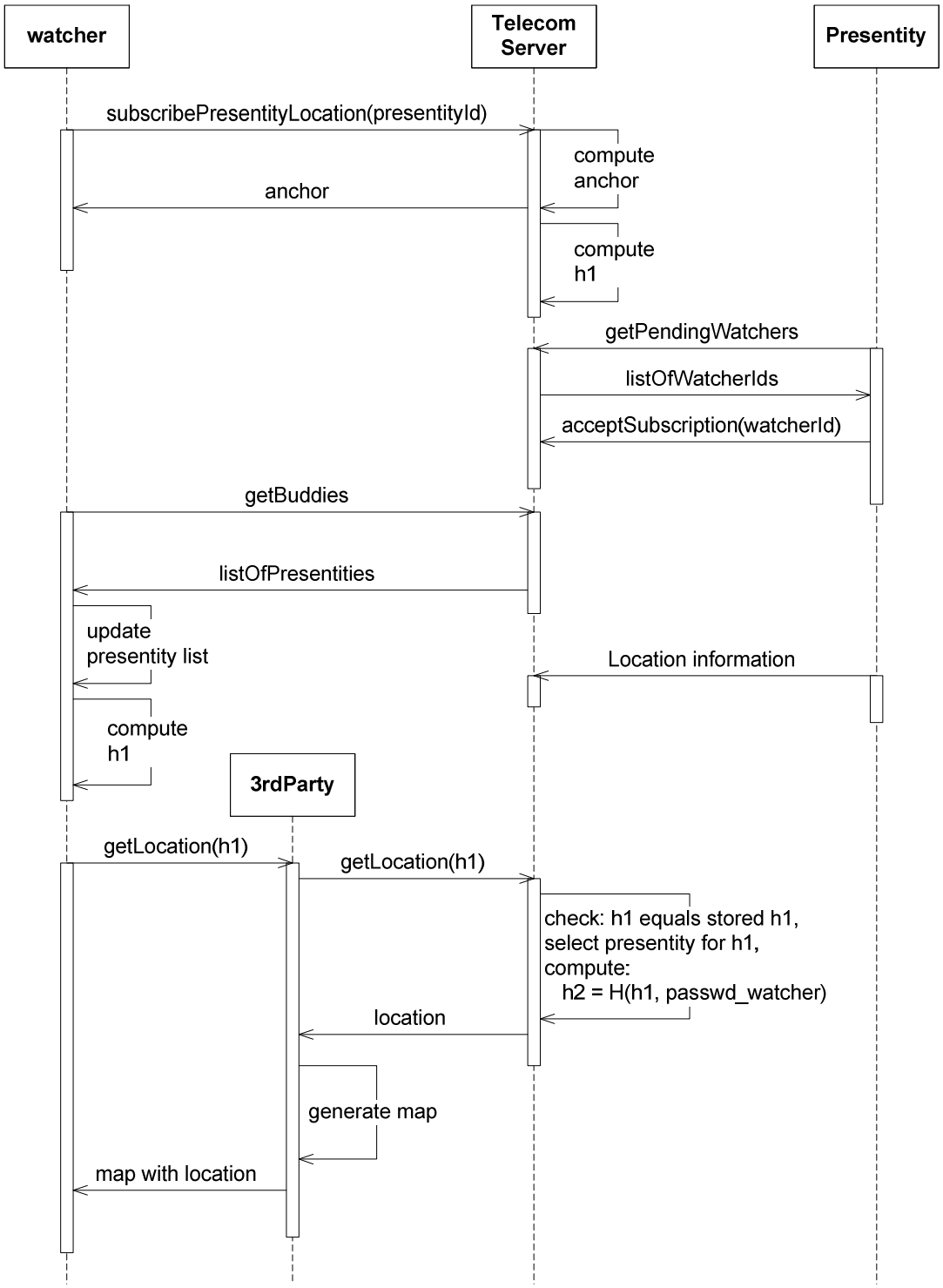


Figure 4-29 Subscription and location query interactions

Finally, the telecom server itself may malfunction. Position requests which cannot be processed in time also lead to different hash values on the client and the telecom server.

To overcome synchronization problems, our prototype system provides a simple mechanism which allows recovery of hash values without further user interaction. If the Telecom Server cannot assign a received hash value to a certain watcher, it returns the coordinates (0, 0) to the client, indicating that the server has processed the request but could not find the matching hash value for whatever reason. Thereupon the client simply sends a new `subscribePresentityLocation()` message to the same presentity. Although the relationship between the

watcher and this particular presentity already exists, a new root is computed. This root value reinitializes the respective hash values stored in the database of the Telecom Server and the watcher is able to request the position again.
Hash chains that are calculated in reverse order of their use are not that susceptible to application faults or to transmission errors. If the one-time-passwords are all calculated and stored before the first use, the chain can never get broken. Even if a single one-time-password gets lost on the way to the telecom server, it can still be checked as valid. The telecom server has to apply the hash function more than once to the received one-time-password and to check if this value matches the last received value of a chain. If the application lacks the memory to store the hash chain in advance, only the index value n of the current hash has to be made persistent and this can be done, before the hash value is used.
The use of hash values does not only allow the implementation of privacy schemes on mobile devices with restricted computing resources, but also increases the efficiency of the Telecom Server implementation. The Telecom Server has to enforce usage policies, that is it has to check if the watcher (user A), who requests location information through third party application X, is authorised to actually locate the presentity (user B). Standardized location APIs (e.g., Parlay X) allow to request the location for a user identifier, but do not contain the original requester identity. Thus a third party application can be allowed to locate a certain group of users, but an explicit relationship between single users cannot be modelled.
The hash value based scheme that is proposed solves the problem of access control and is furthermore very efficient in retrieving data tuples from access control lists. The question, whether a watcher is allowed to locate a presentity is solved during subscription phase. Furthermore the first hash value is calculated from the root value and this hash value also serves as key to store the authorisation data like watcher identity, presentity identity, or location operation. The Telecom Server thus maintains a hashed access control lists, meaning that every authorisation data tuple is indexed by a hash value that guarantees high performance data access. This means that our scheme needs exactly one access to a data base table (or any other access control list repository) in order to resolve authorisation rights, whereas an implementation based on standard APIs would at least have to first fetch the presentity data from a user table and then to look up the watcher (and/or application) in an ‘authorised watcher’ table and to check on the allowed operations. Thus a performance gain for the access control look-up of at least a factor of 2 is realized.
Another advantage of a subscription based authorisation scheme is, that the presentity can easily revoke already given authorisations. The Telecom Server has just to delete the stored hash value from the access control list, which forces a new subscription phase by the time the next localisation requests arrives.

4.4.4 Prototype Realization

In order to validate the architecture and measure the performance of the system, a prototype telecom privacy server was implemented that obtains real location information from a Parlay-X location service. Thus, several mobile phones represent the presentities that can be localized. The watchers are currently implemented on a J2ME Personal Profile 1.0 [J2ME] platform.
A simple ‘Buddy Finder’ application has been implemented for demonstration purposes, that allows to subscribe to location information of another user. If the presentity accepts the subscription, its nickname is added to the watcher’s buddy list and can further on be located. Our

prototype application returns as result a geographical map with a pin representing the user's location (see Figure 4-30).
All interactions between the mobile devices and the telecom server are using SOAP based Web Service interfaces. The Wingfoot Mobile Web Services Platform [Wingfoot] was used to implement the client application on the mobile. Integration with the Apache Axis [Axis] based server was possible without any complications.

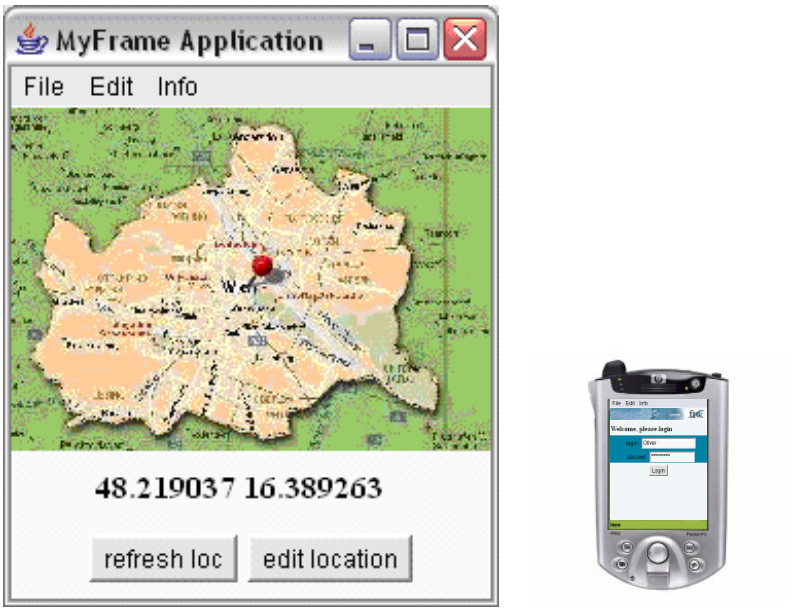


Figure 4-30 'Buddy Finder' application

A second prototype application was developed that concentrated on location information as logical presence data in contrast to geographical coordinates. The application tracks the user and checks periodically if the user enters or exits a certain area, e.g., 'in the office', 'at home'. The scenario is a bit more complex than that described in Figure 4-26: the service starts with the watcher (which is identical with the presentity in this case) sending the application one single `startSession()` request with the pseudonym as parameter. The application requests from the telecom server to be notified with the new position of the user every T minutes (using the pseudonym as parameter). This architecture has the advantage that the application can communicate much more efficiently with the telecom service than the wireless client and it relieves the client from transmitting a high amount of data. When the terminal exits a certain geographical zone, the user is notified and the service session terminates. The logical area information is used to change the user's communication profile, e.g., being reachable for different buddy groups.

4.4.5 Performance and Security Considerations

The performance gain of the hash value based scheme that is proposed is determined by the following operations:

- Hash value calculation at watcher and server
- Check of the authorization of a location request based on the hash-value at the server

One reason for using hash techniques is their computational efficiency that makes them suitable for today's wide spread mobile devices. Our performance measurements were under-

taken for MD5, SHA-1, MD5/HMAC and SHA1/HMAC, carried out on a mobile emulator of the J2ME Wireless Toolkit 2.0 with preset JVM speed emulation of 100 byte-codes/millisecond. All results in Table 4-2 and Table 4-3 are mean values based upon calculation of 100 hash values.

Our performance analysis first concentrated on an implementation of the hash chain scheme that calculates one-time-password in reverse order. Our implementation does not store the calculated hash values, but has to partially rebuild the hash chain for each authentication. Storing the hash values would mean to store n 128 bit (MD5) or 160 bit (SHA-1) values for each buddy, that shall be located. This amount of memory may not be available on today's typical mobile devices. If the hash chain length is set to n=10, the server allows for 10 authentications before the system needs to be reinitialized. The first authentication requires the computation of 10 hash values, the next 9 and so on. This means that the largest delay for getting the hash chain is determined by the first authentication and grows linearly with n. If it is assumed that users are willing to wait at most 5 seconds for the calculation of the first hash, itr can be seen from Table 4-2 that this allows for a hash chain length of 100 hashes for MD5 and a length of 36 hashes for SHA-1.

To keep computing effort low and authentication delays acceptable, n has to be small which however results in frequent re-initializations. It can be seen from the calculations that an implementation based on reverse chained hash calculations (without storing the hash chain once it is computed) is not feasible on mobile devices with low processing power.

The calculation for a forward order hash chain based on the HMAC algorithm that needs a secret key in order to secure the hash chain. Computing the MD5/HMAC and SHA1/HMAC algorithm takes longer than the simple hash functions MD5 and SHA1 (see Table 4-2), but since only one HMAC has to be calculated per authentications and the hash value can be safely calculated from a previous one (n=1), the HMAC procedure is fast enough for our purposes. It can also be seen from Table 4-2 that the HMAC calculation takes about three times as long as the underlying hash function. This means that already for a hash chain length of n=4 the HMAC scheme is faster than the reverse order scheme.

function	mean time
MD5	51 ms
SHA-1	139 ms
MD5/HMAC	164.1 ms (0.164 sec)
SHA1/HMAC	448 ms (0.448 sec)

Table 4-2 Mean time in ms for single hash value calculations

function	Total computation time for 10 authentications	Total computation time for 100 authentications
MD5	2190 ms (2.19 sec)	205003 ms (~3.4 min)
SHA-1	6677 ms (6.677 sec)	701950 ms (~11.7 min)

Table 4-3 Reverse order one-time-password scheme and expected mean time needed for calculation

From a security point of view MD5 or SHA-1 as the underlying algorithms of HMAC can be considered as secure enough given the (not so high) sensitivity of the users's current location data. On the one hand if higher secrecy of data is required, SHA1 should be preferred over

MD5 since collisions in compressing functions have been found for MD5 [Dobbertin]. If processing power is critical, MD5 will be the better choice because it is computed approximately three times faster than SHA-1.
In general, it is not possible for an intruder to calculate easily hash values by eavesdropping the previous hash value. The calculated HMAC hash value is the result of the concatenation of two parts, the previous hash value and the shared secret. Thus the reproduction of the resulting HMAC hash value is not possible even if one knows the previous hash value.
Specifically, examinations on HMAC in [Bellare] show that finding a collision (guessing the input values that result in the same hash value) for hash values of l = 128 bit length would require $2^{l/2}$ messages for a given key. It can be assumed that this is an improbable event to occur. As [Bellare] further states, for 2^{64} using the same key an attack would require approximately 250.000 years on a 1 Gbit/s link.

4.4.6 Conclusion

A scheme has been proposed to protect localized users from being identified by third party applications that need only to process their location, but not their identity. The main advantages of such a system are:

1. A third party application cannot monitor or build a location (movement) profile of the presentity because the pseudonyms change for every location request.
2. Each third party application would receive different pseudonyms for the same presentity, so that profiles across different applications cannot be aggregated.
3. Since the presentity trusts the watcher (and grants the subscription to his/her location information), no additional security responsibility has to be taken by the LBS to check the authorization of the watcher. Look-up performance is increased and access control is simplified by using hash values as keys in access control lists of the telecom server.
4. The HMAC algorithm has been identified as feasible to implement hash chains on a mobile device. The chosen mechanism to compute and validate hash values represents a combination of the two existing techniques HMAC and one-time-password mechanism based on reverse hash chains. It combines the advantages of both: high cryptographic security despite acceptable computational effort.

4.5 A Terminal-Based Location Enabler for the IP Multimedia Subsystem

This chapter describes the system architecture for a terminal-based location service enabler in the Third Generation (3G) IP Multimedia Subsystem (IMS). The idea to integrate a terminal-based location with the IMS was developed during the P4 project "Services in the IP Multimedia Subsystem - SIMS" by our research group at the Telecommunications Research Center Vienna [FTW] and the results of this research were published in several papers [Pailer 05], [Pailer 06], [Fabini], [Reichl 06a], [Reichl 06b], [Gartner].

A location service enabler provides data about the geographical position of mobile terminals to requesting clients. Since location data can be regarded as a type of presence information, an architecture that builds on the IMS presence specifications is proposed. It is shown that several mechanisms for processing presence like notification handling, access control and privacy management apply to location data as well. It is therefore possible to reuse a large part of the IMS presence infrastructure.

Based on the observation that the processing of trigger conditions for location information and creating notification messages should be done in the terminal itself, a terminal-based location enabler is proposed. The terminal uses a built-in method to determine its geographical position such as a Global Positioning System (GPS) module. It is argued that the proposed architecture scales better and is more accurate, efficient and cost effective than current network-based location enablers.

4.5.1 Introduction

Location Based Services (LBS) provide added value mainly by using the physical position of mobile users. Location data may consist of plain geographical coordinates, access point cell IDs, civil location in form of postal addresses or more abstract definitions like 'in the office', 'at home'. Examples of services are a map showing the user's current location or changing the handling of incoming calls when the user enters a specified area.

Service enablers, defined to expose network functionality to external service providers, are becoming the cornerstones of modern service architectures defined by the Parlay Group, the Open Mobile Alliance (OMA) or in the IP Multimedia Subsystem (IMS). A Location Service Enabler is a functional entity in the network enabling value-added services to query the current position of a user or to request a trigger when a specified area is entered or left. Current standardization in the 3rd Generation Partnership Project (3GPP) [3GPP TS 22.071] [3GPP TS 23.271] concentrates on Location Service Enablers that use core network components to provide location data of a user's terminal.

The starting point of the proposed handset-based architecture is the observation that location information should be processed at its source, at the user terminal itself. For advanced notification-based location processing, it will be shown that the proposed distributed, handset-based architecture scales better, is more accurate, efficient and cost effective.

Location information provides considerable value to information and communication services. On the other hand, users are concerned about revealing their position data to others, especially to un-trusted third party applications. Furthermore, most countries have legal restrictions that regulate processing of personal data and the protection of privacy in electronic communications. It is of utmost importance that the users can control who gets access to their

location data and that the transport in the network of such sensitive data is protected by strong security mechanisms.
The target of this service is the IP Multimedia Subsystem (IMS) [Camarillo2006], an overlay network on top of the UMTS packet service (GPRS). It has been specified by the 3GPP in the last years and is currently in the rollout phase. The IMS makes heavy use of the Session Initiation Protocol (SIP) and its extensions and defines several service enablers such as presence, instant messaging, push to talk. Whereas most UMTS applications can be realized without the IMS the specified mechanisms promise a uniform, standardized way of handling quality of service, charging, roaming and integration of different services.
The analysis of the IMS presence system shows that the requirements regarding access authorization, encryption and privacy for presence are indeed identical to those for location. The concept of presence was introduced in instant messaging systems. Presence is information about the online status of other users. A watcher (user) can subscribe to notifications about the state of another user. The watched user is called 'presentity'. Classical presence information is defined as the willingness of the presentity to communicate.
Presence and geographical location have semantic differences: presence attributes, defined by the Internet Engineering Task Force's (IETF) under "rich presence" (RPID, [RFC 4480]) such as location type, activity, or sphere, can each take a small number of values, whereas geographical location has a continuous range of values, limited only by the location accuracy. This requires different handling of subscriptions: whereas for presence all watchers subscribe to changes that occur to a subset of attributes, for location each watcher application may define different criteria for triggering events. Hence, publishing and storing location information on servers like normal presence information is not useful. The proposed architecture takes these differences into account.
Location and presence have however similarities that attest the approach to reuse the IMS presence architecture [RFC 4079]. Examples are the authorization (access rights) and privacy enhancement mechanisms. Moreover, location like presence can be collected both from the mobile terminal and from the network and, mostly important: this can be done using the SIP/SIMPLE protocol stack.
Based on these similarities, this thesis states that location data can be regarded as a special type of presence information that will be subsequently called *location presence data*. The proposed architecture for an IMS Location Service Enabler is thus an extension of the existing IMS presence system. The proposed system design builds on the request routing, authorization, encryption and privacy mechanisms of the IMS presence enabler, but extends the existing specifications in order to support a distributed terminal-based Location Service Enabler.
The contributions of this chapter are summarized as follows:

- The requirements for location services are identified and typical use cases are described.
- A novel location architecture based on application layer signalling is defined, that supports both terminal generated and network-calculated location information.
- Location services are integrated into IMS by using the SIP protocol (subscribe and notify messages).
- The message overhead is reduced to a minimum (solving the notorious efficiency and scalability problems in GSM networks) by extensive use of event triggering at the terminal. This allows for the first time to support applications that require advanced functionality such as area location notification, or proximity notification.

- In the proposed architecture, access control of the location information can be either distributed (at the terminal), or centralized in the server entities, using the same mechanisms as for presence.

The rest of the chapter is organized as follows: the following section presents an overview of 3G Presence and Location Enablers. Section "Architecture of an IMS Location Enabler" motivates the chosen approach of a terminal-based Location Enabler and highlights the proposed architecture. "Location Presence Data" discusses the integrating of location data with presence information. Considerations about performance and costs are presented in Section "Performance and Cost Benefits". Finally related work is listed.

4.5.2 Requirements and Use Cases of Location Services

Location Based Services (LBS) are services that require information about the physical position of a user. Location data may be plain geographical coordinates, access point cell ids, civil location in form of postal addresses or more abstract definitions like 'in the office', 'at home'. The process of gaining location data will be analyzed in the following use cases. These scenarios show interactions between typical actors in a location enabled network. Actor names have been taken from "A Model for Presence and Instant Messaging" [RFC 2778]:

Presentity

An entity that is present in the network. Presence includes a physical location that can be provided to watchers. Presence information containing physical location will from now on be referred to as location presence data.

Watcher

An entity that observes a presentity and gains location presence data of this presentity. Watchers may be users or applications. An application watcher may act on its own or on behalf of a user.

Presence Service

A Presence Service accepts, stores an distributes presence information. It acts as a presence information broker between watcher and presentity and has to enforce access and privacy policies regarding presence data.

Location Service (LCS)

3GPP defines a Location Service as a network enabler that is able to provide 'LCS capabilities', namely location information of mobile stations [3GPP TS 22.071]. A 3GPP Location Service may be the source of location presence data for a Presence Server.

4.5.2.1 Use Case "Peer-to-peer location request"

Figure 4-31 shows the use case diagram "peer-to-peer location request":

- A watcher requests the physical location of a presentity.
- The request is contained in a SIP SUBSCRIBE message and is sent to the SIP URI of the presentity. The request may be processed by a Presence Server before reaching the presentity.
- The authorization of the watcher is checked by the presentity. A Presence Server may also authorize the request. The presentity shall be able to manage location request authorization by adding or removing watchers from buddy lists.
- The presentity uses a terminal-based location method to determine it's position. If the terminal-based location method is unavailable, a Presence Server may request the location from a network-based location enabler as a fall back method.

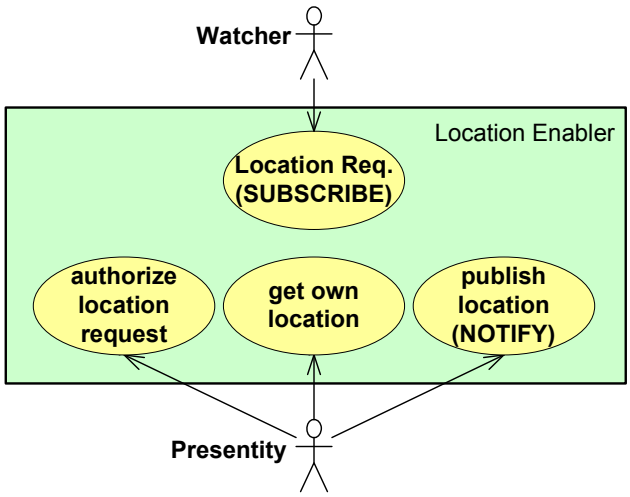


Figure 4-31: Use case diagram "Peer-to-peer location request"

4.5.2.2 Use case "third party location request"

Figure 4-32 shows the use case diagram "third party location request". This use case is a variation of the peer-to-peer use case:

- The third party application acts on behalf of the watcher towards the presentity.
- The watcher authorizes the third party application.
  Authorization is trivial, if the presentity and the watcher are identical, that is if the presentity uses a location service to locate itself.

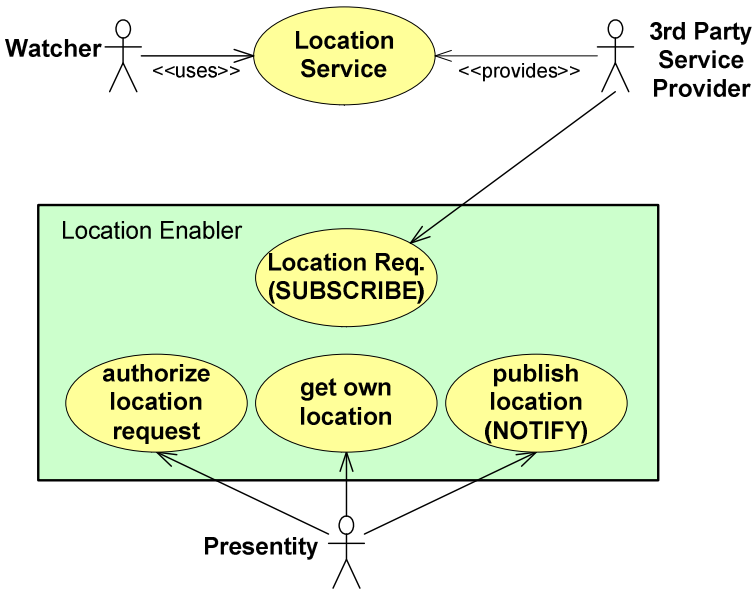


Figure 4-32: Use case diagram "third party location request"

4.5.2.3 Requirements

The following requirements for a location service have been identified:

Efficiency

Location presence data shall be provided by using terminal and network resources as efficient as possible.

Scalability
The proposed architecture shall be usable for a wireless network operator and has thus scale to a magnitude of millions of subscribers.

Privacy/security
Location presence data shall be protected against unauthorized distribution. The presentity has means to manage and control single watchers and groups of watchers dynamically. The presentity should have the possibility to remain anonymous to third party applications.

Support of both GPS and Network calculated location
If the terminal is unable to provide location data (e.g., no GPS module available), a network location enabler shall be used (e.g., GMLC provided coordinates or GSM cell-ID or WLAN access point ID).

Flexibility
The proposed system shall support diverse formats of location data: e.g., geographical coordinates [WGS-84], points and areas, postal addresses or cell-IDs.

Location Modes
The watcher should be able to specify whether location data of the presentity is required only once, periodically or triggered (e.g., when entering a specified area).

Roaming
Scenarios in which the location source (presentity) is in a different network or domain than the watcher should be possible.

IMS/NGN
The architecture shall be in line with the 3GPP IMS specification.

Third party access
Access by third party applications to location data shall be supported.

Costs
Investments per user shall be kept to a reasonable minimum.

4.5.3 IMS Presence Enabler

The presence enabler in the IMS is based on the specifications published by the IETF working group “SIP Instant Messaging and Presence Leveraging Extensions (simple)” [SIMPLE]. [RFC 2778] defines an abstract model for a presence and instant messaging system. The SIMPLE specifications make use of the subscribe-notify mechanism of the SIP event system [RFC 3265. The presence system of the IMS is specified in [3GPP TS 23.141], the presence event package in [RFC 3856].
Figure 4-33 shows all elements of the presence system, including the optional resource list server.

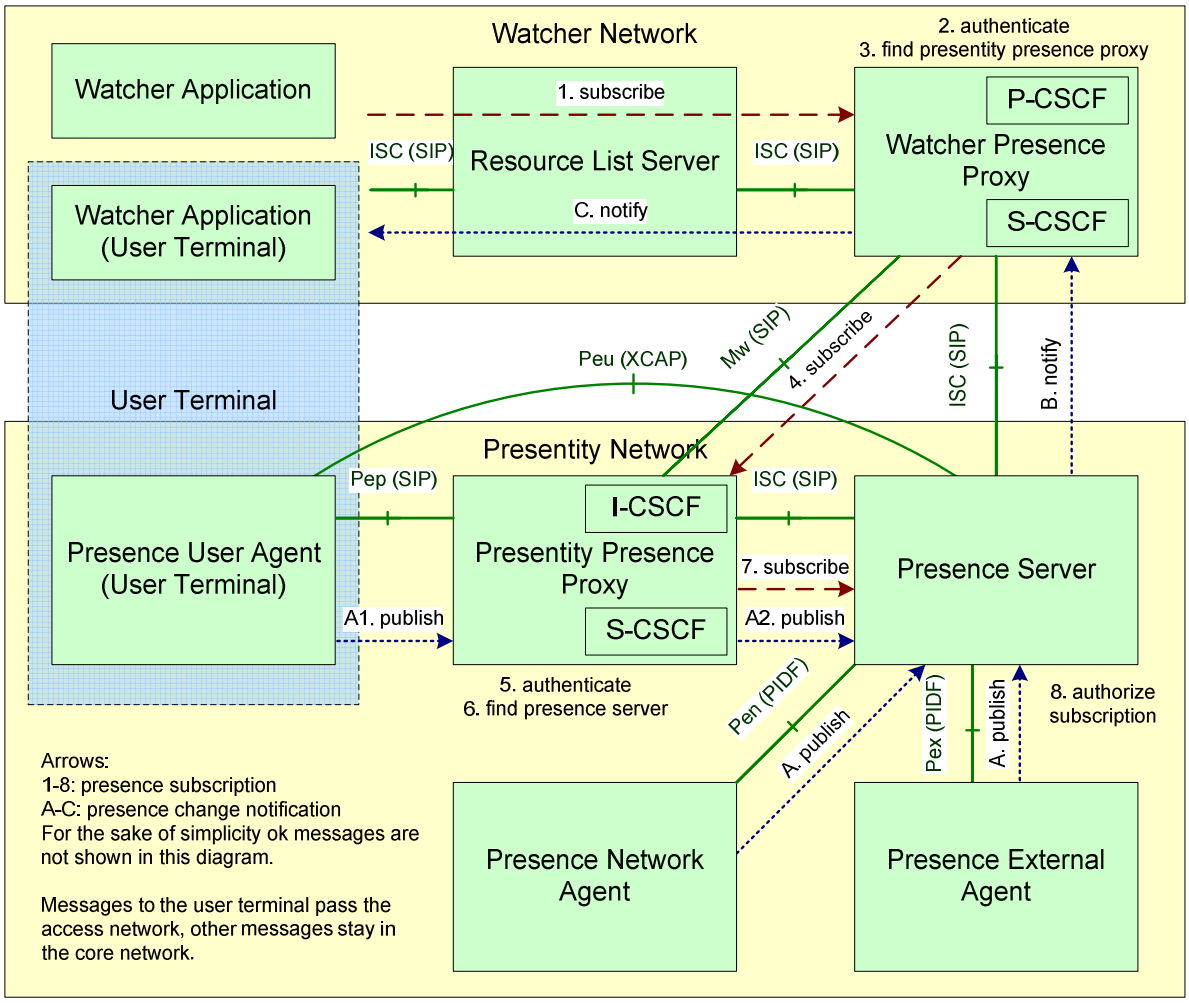


Figure 4-33: IMS Presence System Overview.

The entities have the following functions:

Watcher Application: The watcher application can run on an application server in the core network or on a user terminal in the access network. The application subscribes to the presence information of presentities and receives notifications whenever the presence status changes. The watcher application can supply a notification filter ([RFC 4660], [RFC 4661]) at subscription time to limit traffic or request complex notification behaviour.

Resource List Server: The Resource List Server (RLS, [RFC 4662] and [XCAP LIST]) relieves the watcher's user agent from subscribing to and managing notifications from all addresses on his contact list. Instead, the contact list is stored on the RLS and the PUA subscribes to the contact list and receives bundled notifications. This reduces traffic on the air interface.

Presence User Agent: The Presence User Agent (PUA) sets policies for subscribers (using the XCAP protocol [XCAP] [XCAP DIFF] over HTTP) and sends notifications (using the SIP PUBLISH message) if its presence state changes. Since the presence user agent does not know about the subscriptions to its presence data, it has to publish presence information independently of the actual need for it. Even if there are no watchers at all, the PUA has to publish every single presence change of the user.

Presence Network Agent: The Presence Network Agent supplies presence information coming from the network (e.g., cell based location information). The presence network agent sends PIDF [RFC 3863] or rich presence (RPID, [RFC 4480]) documents to the presence server.

Presence External Agent: The Presence External Agent supplies presence information coming from external sources such as a calendar.

Presence Server: The Presence Server (PS), also known as **Presence Agent** (PA), is the entity in the IMS core network responsible for the presence information of all subscribers to this network. The presence server maintains presence information and sends notifications to watchers. The presence server is responsible for the authorization of all subscriptions.

Watcher Presence Proxy: The Watcher Presence Proxy provides watcher related functions such as authentication of watchers. It is a combination of SIP proxies: P-CSCF (proxy call state control function) and S-CSCF (serving CSCF) in the watcher's network.

Presentity Presence Proxy: The Presentity Presence Proxy provides presentity related functionality such as determining the correct presence server. It is a combination of I-CFCF (interrogating CSCF) and S-CSCF in the presentity's network.

Figure 4-33 illustrates two message flows: a watcher subscribes for presence information (1 to 8) and a notification for new presence information (A to C). The authentication process (2, 5) involves the HSS (Home Subscriber Server) which is not shown in this diagram.

4.5.4 Location Service Enabler

Current 2G and 3G networks use a network-based Location Service Enabler to provide location data to location applications. 3GPP specification [3GPP TS 22.071] gives a general description of location services (LCS) and service requirements for third generation networks. [3GPP TS 23.271] specifies the mechanisms to support mobile location services for operators, subscribers and third party service providers. The radio signals of the wireless network are used to determine the (geographic) location of the user equipment (UE). The specification explicitly assumes that "positioning methods are Access Network specific, although commonalties should be encouraged between Access Networks".

Interworking with the IMS has been introduced in version 6.7.0 for UMTS release 6 in March 2004. The central component in the 3GPP LCS design is the Gateway Mobile Location Center (GMLC) that offers location services. Figure 4-34 shows the localization of an IMS public user identity:

1. A client requests the current location of a public IMS identity (SIP-URI) from the GMLC in the requesting network (R-GMLC).
2. The R-GMLC forwards the request to a Location IMS Interworking Function in the requesting network (R-LIMS-IWF).
3. The R-LIMS-IWF determines the LIMS-IWF in the home network (H-LIMS-IWF) of the target.
4. The H-LIMS-IWF queries the HSS for the Mobile Station International Subscriber Directory Number (MSISDN) of the target SIP-URI and retrieves a routing for the GMLC in the home network (H_GMLC).
5. The H-LIMS-IWF forwards the request to the H-GMLC.
6. The H-GMLC performs a privacy check.

7. The H-GMLC requests a routing towards the GMLC in the visited network (V-GMLC) from the HSS.
8. The H-GMLC forwards the request to the V-GMLC.
9. The V-GMLC queries the HSS in the visited network for the location area of the target.
10. The radio network uses a positioning method to locate the target. The response message takes the route V-GMLC, H-GMLC, H-LIMS-IWF, R-LIMS-IWF, R-GMLC to the client.

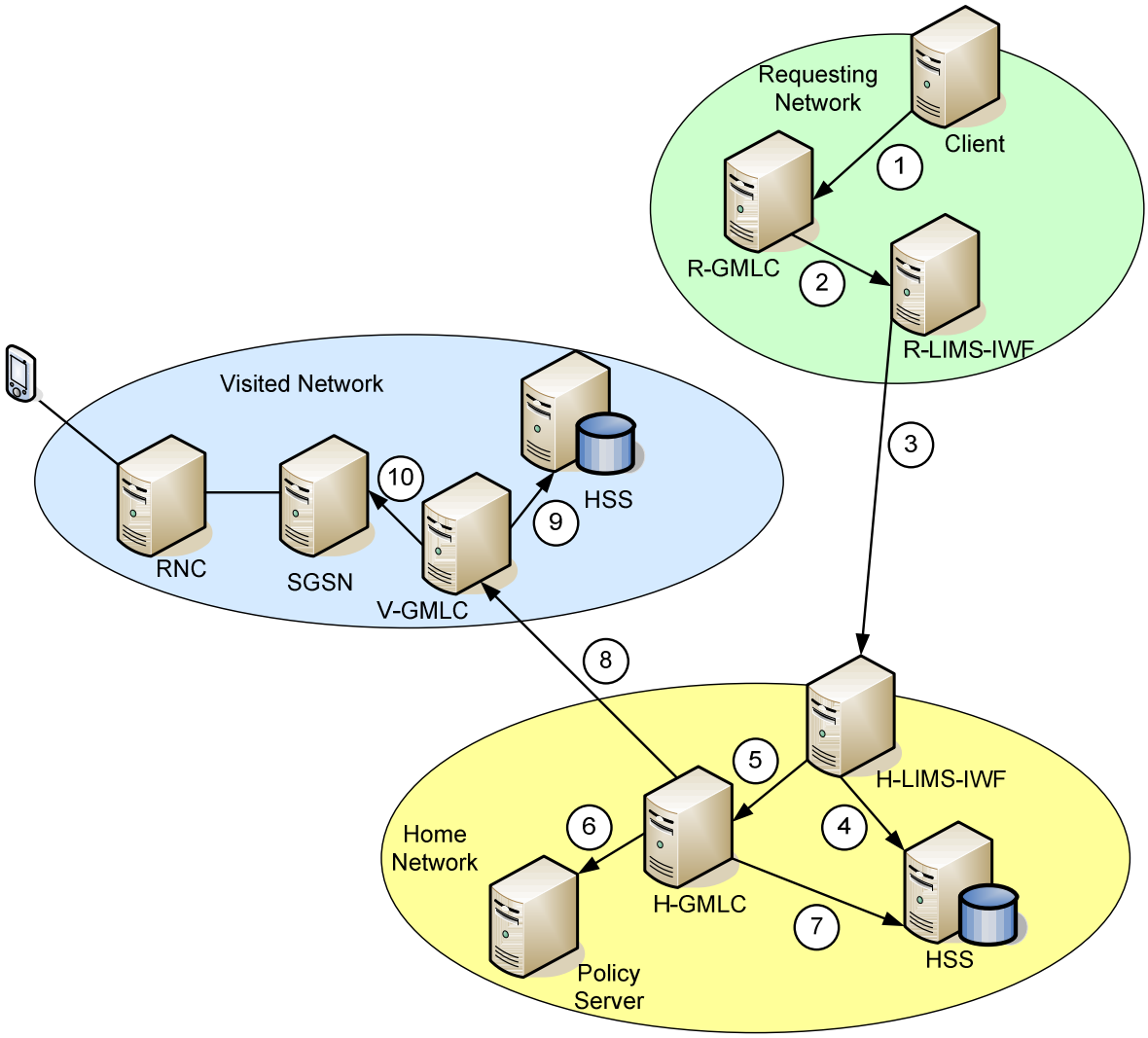


Figure 4-34: Mobile Terminal-Location Request (MT-LR) procedure for IMS Public User Identities in the 3G core network

The following network-based positioning methods are specified in the 3G documents:

Cell Coverage Based. The cell ID is either known to the radio network or can be obtained by paging the terminal. The accuracy of this method depends on the cell size and is typically in the range of 300 m in urban areas.

Idle Period Downlink - Observed Time Difference Of Arrival (IPDL-OTDOA). This method measures the relative time of arrival of pilot signals from different base stations. At least three stations have to be visible to calculate a location. Network planning however optimizes a cellular network for available bandwidth which means reducing unnecessary overlapping of cells [Johnson]. Thus the accuracy of the IPDL-

OTDOA method is not satisfying in real networks (about 50 m – 150 m) and the technology requires a substantial up-front investment in the radio access network.

Network Assisted GPS (A-GPS). The network sends assistance data to a GPS receiver in the terminal in order to speed up the position calculation and to provide service in areas where GPS signals are weak (e.g., inside buildings). Assistance data contains precise GPS satellite orbit and clock information, initial position and satellite selection. A-GPS offers good position accuracy (5 m) but requires a GPS assistance service and necessitates a medium up-front investment in the 3G network. Furthermore A-GPS enabled terminals are not available on the market yet.

4.5.5 Architecture of an IMS Location Enabler

The proposed system architecture is based on the following two assumptions:

1. The optimal source for geographical location data is the user's terminal which is equipped with a GPS module.
2. Geographical location is a special type of presence information that is referred to as call location presence data.

4.5.5.1 Location and Presence

The fundamental assumption that leads to the reuse of the presence architecture is that location information is regarded as a kind of presence information. The location of a user may be related to his presence state. The access to both presence and location data has the same requirements regarding security and privacy. The next sections will show how to re-use the well specified presence system of the IETF (and thus also of the IMS), in particular the authorization, subscription and privacy mechanisms. Location presence data fits smoothly into the existing presence system which can be extended easily with the new location data type.

4.5.5.2 Location Data and Notification Filters

Notification filters enable more complex notifications than pure location updates. The filter is sent within the SUBSCRIBE message from the watcher to the presentity. For example the filter "(longitude TO centre BY distance) OR (latitude TO centre BY distance)" will send a notification whenever the presentity enters a rectangle around the centre coordinates. This reduces traffic over the radio interface (single notification instead of constant location updates) at the price of added complexity at the presentity's terminal. A format for notification filters has been proposed recently within the IETF GEOPRIV working group [Mahy]. The use of the Geography Markup Language (GML) [GML] is proposed to include location information in PIDF documents. Since both languages are XML based, GML elements can be integrated in PIDF documents easily.

4.5.5.3 Terminal Based Location Information

There are several reasons for retrieving geographical location data from the user's terminal. Current network based location methods do not offer satisfying accuracy. Increasing the accuracy of network based location enablers requires costly investments in the core network infrastructure. On the other hand, only a part of all users will use location based services. A terminal based location enabler, such as a built-in GPS sensor or a Bluetooth GPS module, is a relatively inexpensive hardware feature and allows upgrading only those terminals whose users subscribe to or want to use location enabled services. A terminal based location enabler can be accessed in a peer-to-peer mode with minimal impact on network resources, thus making a highly scalable solution possible. Since the proposed solution is totally independent of

the underlying network technology or the network provider, it enables vertical handover (e.g., UMTS - WLAN) and works in roaming scenarios.

4.5.5.4 Proposed System Architecture

In order to access a terminal based location enabler, it is proposed to install a SIP Edge Presence Server at the GPS enabled user terminal. An edge presence server is a presence agent that is co-located with a Presence User Agent (PUA). Since the PUA manipulates the presence information, the edge presence server can be aware of it ([RFC 3856], chapter 3).

4.5.5.4.1 Peer-to-peer location

The simplest form of the proposed system architecture, a peer-to-peer location service, is sketched in Figure 4-35.. The user's terminal hosts an edge presence server for location presence information. It allows third party applications or user terminals to subscribe to location presence data notifications. A watcher application may use the SIP protocol directly or other third party interfaces like the ParlayX location API [PaylayX Loc] to request presence location data. The client software in the terminal of the presentity uses the J2ME Location API [JSR-179] to access the GPS module providing the physical location.

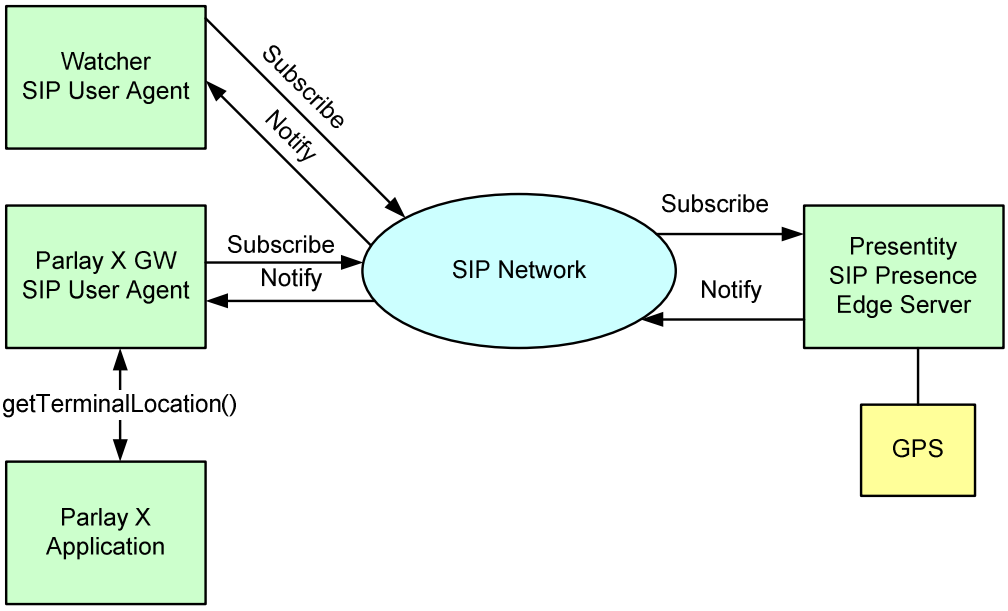


Figure 4-35: Peer-to-Peer System Architecture

While it requires little infrastructure, the peer to peer architecture has two drawbacks: lots of notifications pass the radio network and network based location for terminals without a GPS receiver is not supported.

Figure 4-36 shows the system architecture in server mode. A watcher's subscription is handled by a SIP Presence Server. The Presence Server notifies the presentity about a pending subscription with a watcher info event. Authorization of the watcher can be provisioned by the presentity using the XCAP protocol. Location presence data is requested by the Presence Server with an own Subscribe/Notify dialogue towards the presentity's SIP Edge Presence Server that is co-located with the presentity's user agent. The SIP presence Server acts as a Back-to_Back User Agent (B2BUA)

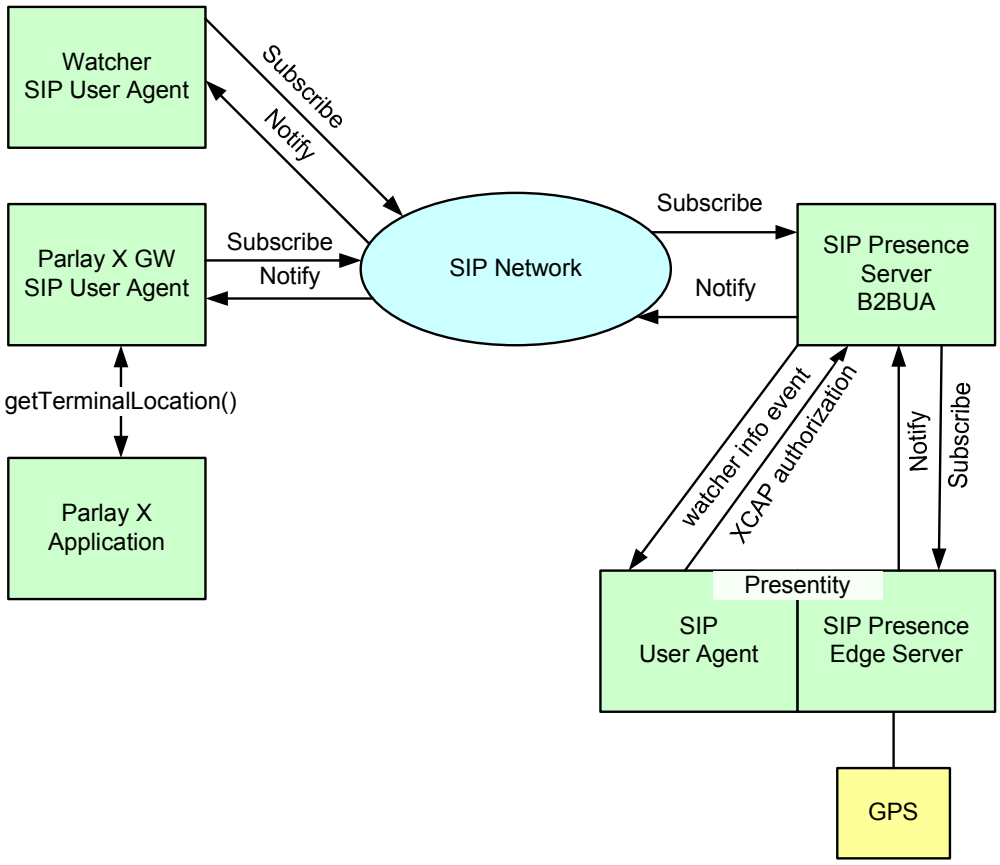


Figure 4-36: Server mode system architecture

4.5.5.4.2 Watcher List Server

To reduce the radio bandwidth required by the edge presence server, a new component, the Watcher List Server (WLS, see Figure 4-37), is introduced. This element of the presence system manages lists of watchers subscribed to the same location events. The edge presence server at the user terminal has to implement the u_t interface (XCAP) to the WLS to configure these lists. Once the watcher list has been configured, any presence notification sent to it will reach all the watchers on the list (arrows b2 and c2 in Figure 4-37).

The improvement achieved by a WLS is based on the assumption that the delivery conditions for location are the same for all the watchers. However, this is not true for all requests. For example one watcher might request periodic notifications while another wants to know when the presentity reaches a specified area. In these cases direct notifications like in the peer-to-peer mode are used (b3 in Figure 4).

The watcher list server is transparent to the watcher and to the rest of the presence system. The only entity affected is the edge presence server on the terminal.

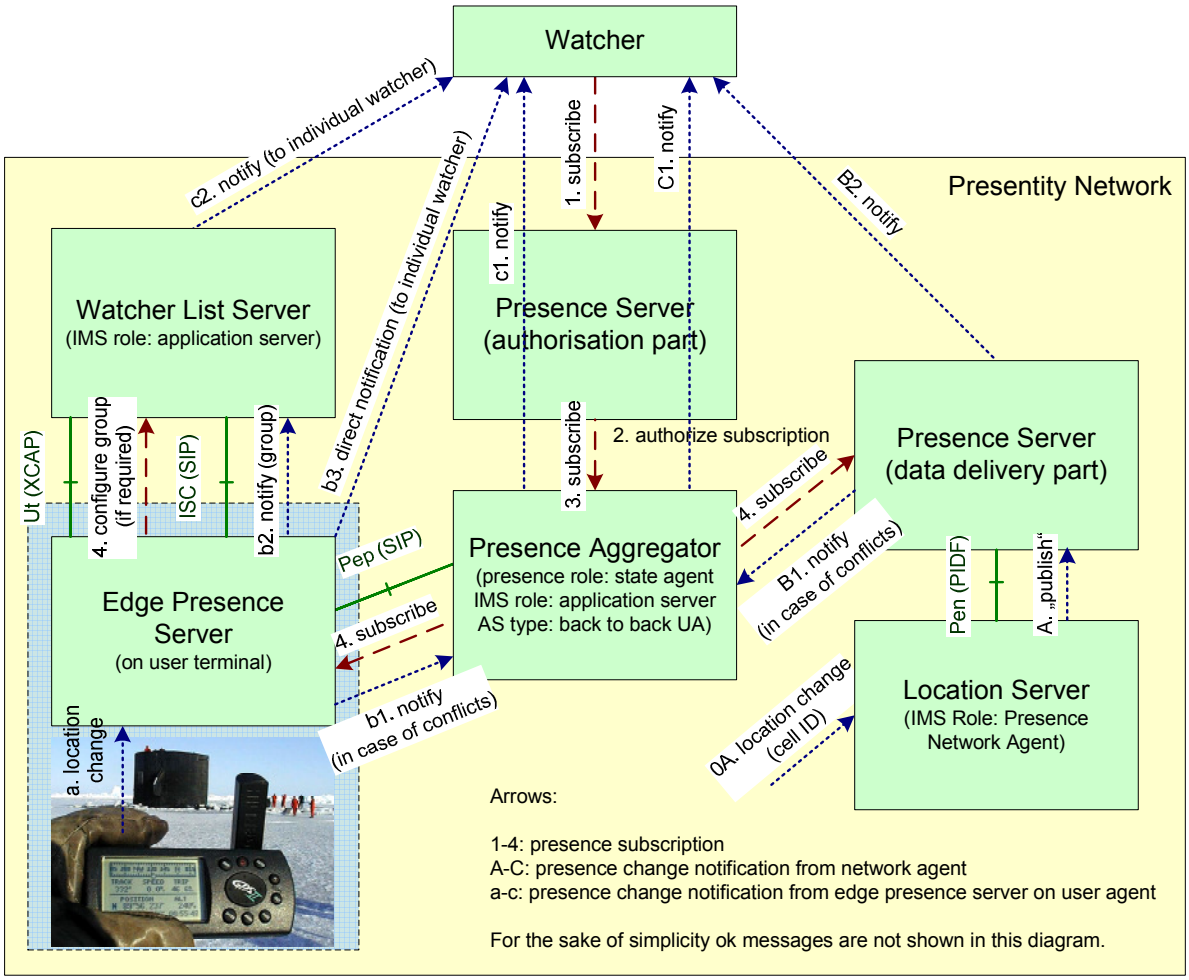


Figure 4-37: Full System Architecture[1].

4.5.5.4.3 Presence Aggregator

The main remaining problems are the support of network based location for legacy terminals and the consolidation of information if both options (GPS and network) are available. Note that the presentity is the only entity able to resolve these conflicts ([RFC 3856], chapter 6.9).

This problem is resolved by the Presence Aggregator. It intercepts all subscription messages. Based on a user data base or on subscription policies, it decides how to locate the presentity. If the location is only available in the network the subscription is forwarded to the presence server. If the user has an active GPS receiver, the subscription will be sent to the terminal. If both options are available the aggregator can check the data for reliability. (Are the GPS co-ordinates within the cell?) Arrows B1 and b1 in Figure 4-37 show the (potentially conflicting) notifications to the aggregator, which forwards only one of them (C1 OR c1) to the watcher.

Like the WLS, the aggregator is transparent to the watcher. It is also transparent to the presentity and all proxies. The presence server authorizes all subscriptions.

It can be assumed that the aggregation function is implemented within the presence server in the core network, especially since the aggregation task takes place after authorization, but before data can be fetched from any presence server.

The resulting architecture relieves the terminal from publishing its location independently of actual watchers. Still, it is compatible with the IMS. It supports both terminal generated and

[1] D. Chayes, Columbia University: photo of GPS receiver

server calculated location keeping the traffic in the radio access network low. Note that the proposed architecture can be applied to any presence data originating at the user terminal.

4.5.5.5 Routing and Addressing

Location presence data shows some essential differences from other presence data. First, location data is semantically different from classic presence data due to the almost continuous state space. Second, location data originates from different sources than normal presence state. As location data is calculated in the presentity's terminal it requires a distributed approach.

In the IETF presence specification "A Presence Event Package for the Session Initiation Protocol (SIP)", [RFC 3856] and "Session Initiation Protocol (SIP)-Specific Event Notification", [RFC 3265] the following components of a presence system are defined. Highlighted statements are important for the proposed architecture and thus for the routing of SIP presence messages containing presence location data:

Presence Agent (PA)

"A presence agent is a SIP user agent which is **capable of receiving SUBSCRIBE** requests, responding to them, and **generating notifications of changes in presence state**. A presence agent must have knowledge of the presence state of a presentity. This means that it must have access to presence data manipulated by Presence User Agents (PUA) for the presentity. One way to do this is by co-locating the PA with the proxy/registrar. Another way is **to co-locate the PA with the presence user agent** (PUA)of the presentity. However, these are not the only ways, and this specification makes no recommendations about where the PA function should be located. A PA is always addressable with a SIP URI that uniquely identifies the presentity (i.e., sip:joe@example.com). There can be **multiple PAs** for a particular presentity, each of which **handles some subset of the total subscriptions** currently active for the presentity. A PA is also a **notifier** (defined in RFC 3265) that supports the presence event package."

[RFC 3856], chapter 3

Presence Server

"A presence server is a physical entity that **can act as either a presence agent or as a proxy** server for SUBSCRIBE requests. When acting as a PA, it is aware of the presence information of the presentity through some protocol means. When acting as a proxy, the SUBSCRIBE requests are proxied to another entity that may act as a PA."

[RFC 3856], chapter 3

Edge Presence Server

"An edge presence server is a **presence agent that is co-located with a PUA**. It is aware of the presence information of the presentity because it is co-located with the entity that manipulates this presence information."

[RFC 3856], chapter 3

State Agent

A state agent is a **notifier** which publishes state information on behalf of a resource; in order to do so, it may need to **gather such state information from multiple sources**. State agents always have complete state information for the resource for which they are creating notifications."

[RFC 3265], chapter 2

"State agents are core to the presence event package. **Whenever the PA is not co-located with the PUA for the presentity, the PA is acting as a state agent**. It collects presence state from the PUA, and aggregates it into a presence document. Because there can be multiple PUA, a **centralized state agent is needed to perform this aggregation.** That is why state agents are fundamental to presence. Indeed, they have a specific term that describes them - a **presence server**."

[RFC 3856], chapter 6.11

Handling of Forked Requests

"The presence event package specification **only allows a single dialog to be constructed as a result of emitting an initial SUBSCRIBE** request. This guarantees that only a single PA is generating notifications for a particular subscription to a particular presentity. The result of this is that a presentity can have multiple PAs active, but these should be homogeneous, so that each can generate the same set of notifications for the presentity. Supporting heterogeneous PAs, each of which generates notifications for a subset of the presence data, is complex and difficult to manage. Doing so would require the subscriber to act as the aggregator for presence data. This aggregation function can only reasonably be performed by agents representing the presentity. **Therefore, if aggregation is needed, it MUST be done in a PA representing the presentity**."

[RFC 3856], chapter 6.9

Subscription Migration

"Subscription migration is the act of **moving a subscription from one notifier to another notifier**."

[RFC 3265], chapter 2

The proposed terminal-based location enabler architecture assumes, that the terminal is equipped with some kind of locating capability (e.g., a GPS module). This means that the presence agent for location presence data is located in the terminal and thus is co-located with the presence user agent. It follows from the definitions above that this presence agent for location presence data is in fact an edge presence server. It is the task of this **Location Presence Agent** (LPA) to process SUBSCRIBE requests for location presence data and to provide location information in NOTIFY messages. As the LPA is a software component running in the user's terminal, direct access to location information (from the GPS module) is possible.

Furthermore the proposed architecture shall be compatible to network-based location enablers (e.g., a cell-id based method in GSM networks). This means that there may be a presence agent, that is NOT co-located with the presence user agent, thus acting as a state agent. The definitions above mandate, that there is a centralized state agent, a presence server, that represents the presentity by forking subscriptions to the edge presence server and the state agent and by aggregating location presence data from an edge presence server and the state agent.

The citations from the presence specifications above prove that the described architectural concept for location data presence handling is indeed compatible with a standard presence architecture. There is however the difference that subscriptions for location presence data may (most probably) be handled by different components than classical presence data. It is therefore necessary to define a SIP message routing mechanism that is able to distinguish between subscriptions for classical presence and location data presence. Once a SIP dialog has been established by an initial SUBSCRIBE message, all subsequent SIP requests will take the route specified by that dialog. The following possibilities were analysed in the course of the FTW research project:

Routing based on subscription filters:

The initial SUBCRIBE message for location presence data is directed to the SIP URI of the presentity and contains some kind of subscription filter document in the request body, that specifies that the watcher is only interested in location presence data. A presence server that represents the presentity detects that the watcher is only interested in a subset of all possible subscriptions and forwards the request to the responsible presence agents (e.g., an edge presence server and/or state agent for location presence data).

A disadvantage of this solution is that there must be a presence server that routes all subscriptions, for classical presence as well as location presence.

Routing based on *locpres:* URI:

This solution proposes to separate subscriptions and notifications for location presence data from classic presence by introducing a new *locpres:* URI-scheme. The intended usage of the locpres: URI follows closely the usage of the pres: URI. However, it allows routing a subscription for presence location data to the edge presence server in the presentity's terminal or to a state agent representing the presentity by addressing the locpres: URI of the target.

A disadvantage of this solution is that it may be difficult and cumbersome for the user to distinguish between too many URI schemes and the semantics behind those schemes.

Routing based on *locpres* Event header:

This solution proposes to define a *locpres* event package for SIP as a concrete instantiation of the general event notification framework defined in RFC 3265. SIP routing of initial SUBCRIBE messages can be based on the content "locpres" of the SIP Event header field.

This solution seems to integrate well in the existing SIP presence framework and general SIP specifications.

Presence user agents can for example use the "Watcher Information Event Template-Package" [RFC 3857]. This means that the PUA can subscribe to "locpres.winfo" notifications in order to be notified about changes in the subscription state regarding *locpres* events.

Furthermore RFC 3840 introduces "User Agent Capabilities" in form of feature tags that can be conveyed in form of parameters in the Contact header field to other user agents and registrars. In particular a Contact header with the sip.event media tag containing a value of "locpres" can be used by a PUA to indicate that subscriptions for location presence data shall be routed to this address. Sip.event feature tag parameters are also used in RFC 3841 that defines the notion of "caller preferences" about request handling in servers. These preferences include the ability to select which Uniform Resource Identifiers (URI) a request gets routed to, and to specify certain request handling directives in proxies and redirect servers. It does so by defining three new request header fields, Accept-Contact, Reject-Contact, and Request-Disposition, which specify the caller's preferences. Again a value of "locpres" in a sip.event media feature tag can be used by a presence server to route a subscription for location presence data to the responsible edge presence server or state agent.

Considering advantages and disadvantages of above solutions, it is proposed to specify a *locpres* event package for the handling of location presence data. An IETF Internet-Draft

specifying a *locpres* event package has been published by our research group (see Appendix A: Location Presence Internet Draft).

4.5.6 Location Presence Data

Location presence data refers to the physical location of a user's terminal as a special type of presence information. IMS supports presence based on SIP extensions and mapping of location presence data to IMS protocols seems straight forward, but has not been described comprehensively in the standard specifications. This chapter defines an 'IMS profile' for making location presence data available to users and applications in the IMS by specifying how the different standard and draft documents should be used in this context. Furthermore some extensions to existing specifications are proposed in order to support all features of widely-used third party location APIs like the ETSI Parlay X Terminal Location Service [Parlay X Loc] or the OMA Mobile Location Protocol [OMA MLP].

4.5.6.1 IMS Presence

The IMS functionality is largely based on the SIP protocol [SIP]. The IMS presence service is specified in RFC 3856, where the SIP protocol is used to transport subscriptions and notifications of user presence. SIP presence is based on the general presence service model and definition of presence terminology in RFC 2778. A Presence Service accepts, stores and distributes presence information about users, called presentities. Interested parties, called watchers, can access presence information at the presence service.

SIP presence defines an event package according to the general event notification framework for SIP RFC 3265. A watcher uses the SIP SUBSCRIBE method to request notifications about a user's presence at the presence server. The presence server uses the SIP NOTIFY method to convey presence information about a presentity to the watcher.

Presence historically only included the user's status 'online' or 'offline', but has developed into a broader concept since then. The minimal presence information content is defined in RFC 2778 as one or more PRESENCE TUPLES, containing a STATUS, an optional COMMUNICATION ADDRESS and optional OTHER PRESENCE MARKUP. STATUS has at least the mutually exclusive values OPEN and CLOSED that show the willingness of the presentity to accept instant messaging communication. There may be other values of STATUS that do not imply anything about INSTANT MESSAGE acceptance, that may be combined with OPEN and CLOSED.

Presence information is transported in the NOTIFY message body. All SIP presence implementations have to support the Presence Information Data Format PIDF [RFC 3863], using XML to encode presence information. A simple example PIDF document is shown in Figure 4-38. PIDF is a concrete implementation of PRESENCE TUPLES defined in RFC 2778.

```
<?xml version="1.0" encoding="UTF-8"?>
 <presence xmlns="urn:ietf:params:xml:ns:pidf"
     entity="pres:someone@example.com">
   <tuple id="sg89ae">
     <status>
       <basic>open</basic>
     </status>
     <contact priority="0.8">tel:+09012345678</contact>
   </tuple>
 </presence>
```

Figure 4-38: Presence Information Data Format (PIDF) example

4.5.6.2 Location Data in Presence Documents

The IETF GEOPRIV working group deals with privacy and security aspects of transferring geographic information by location enabled applications. [RFC 4119] “A Presence-based GEOPRIV Location Object Format” identifies the need to convey geographical location information within an object that includes a user's privacy and disclosure preferences and which is protected by strong cryptographic security. The specification observes that a presence system offers exactly the wanted level of privacy and security und thus the Presence Information Data Format (PIDF) is proposed to transfer location data. The XML Schema proposed in this specification extends the 'status' element of PIDF with a complex element called <geopriv>. There are two mandatory sub-elements that are encapsulated within <geopriv>. The <location-info> element for geographical information and the <usage-rules> element to specify privacy and security requirements. The <location-info> element can contain any XML schema that is suitable for a particular application, but all implementations have to support the ‘feature.xsd’ XML schema of the Geography Mark-up Language version 3.0 (GML 3.0) [GML].

```
<?xml version="1.0" encoding="UTF-8"?>
 <presence xmlns="urn:ietf:params:xml:ns:pidf"
    xmlns:gp="urn:ietf:params:xml:ns:pidf:geopriv10"
    xmlns:gml="urn:opengis:specification:gml:schema-xsd:feature:v3.0"
    entity="pres:geotarget@example.com">
  <tuple id="sg89ae">
   <status>
    <gp:geopriv>
      <gp:location-info>
        <gml:location>
          <gml:Point gml:id="point1" srsName="epsg:4326">
            <gml:coordinates>37:46:30N 122:25:10W</gml:coordinates>
          </gml:Point>
         </gml:location>
      </gp:location-info>
      <gp:usage-rules>
        <gp:retransmission-allowed>no</gp:retransmission-allowed>
        <gp:retention-expiry>2003-06-23T04:57:29Z</gp:retention-expiry>
      </gp:usage-rules>
    </gp:geopriv>
   </status>
   <timestamp>2003-06-22T20:57:29Z</timestamp>
  </tuple>
 </presence>
```

Figure 4-39: PIDF extension by location information and usage rules

The Geography Markup Language (GML) is an extraordinarily thorough and versatile system for modeling all manner of geographic object types, topologies, metadata, coordinate reference systems and units of measurement. The simplest package for GML supporting location information is the 'feature.xsd' schema. Although 'feature.xsd' can express complicated geographical concepts, it requires very little markup to provide basic coordinates points for the most common use cases (see example in Figure 4-39).

GML 3.0 allows for example to define a surface by specifying the edges of a polygon shape (see Figure 4-40). Thus a presentity is able to send updates that refer to an area rather than a point.

```
<gml:extentOf>
  <gml:Polygon>
    <gml:outerBoundaryIs>
      <gml:LinearRing>
        <gml:coordinates>0,0</gml:coordinates>
        <gml:coordinates>50,0</gml:coordinates>
        <gml:coordinates>50,40</gml:coordinates>
        <gml:coordinates>0,0</gml:coordinates>
      </gml:LinearRing>
    </gml:outerBoundaryIs>
  </gml:Polygon>
</gml:extentOf>
```

Figure 4-40: Polygon shape in GML 3.0

4.5.6.2.1 GML 3.1

In Feburuary 2004, version 3.1.0 of GML was published by the Open GIS Consortium. This version deprecates the use of the gml:coordinates element and recommends the use of the gml:pos element instead. [Winterbottom] recommends the usage of the gml:pos element inside PIDF documents.

It is therefore proposed to use the 'feature.xsd' XML schema of GML 3.1.0 to transport geographical coordinates in PIDF documents. Figure 4-41 shows a geographical position coded according to GML 3.1.0. The gml:pos element has a structured format, in that each field is represented as a double, and a single space exists between each field. The attribute srsName references a well known Coordinate Reference system (CRS) according to ISO 19111. The URN urn:ogc:def:crs:EPSG:6.6:4326 leads to a definition of a well-known CRS specified by the European Petroleum Survey Group that uses latitude/longitude and is equal to WGS84 [WGS-84]. The format of the gml:pos element is thus latitude followed by longitude. A positive value for latitude represents a location north of the equator while a negative value represents a location south of the equator. Similarly for longitude, a positive value represents a location east of Greenwich, while a negative value represents a location west of Greenwich.

```
<gml:location>
  <gml:Point gml:id="point1" srsName="urn:ogc:def:crs:EPSG:6.6:4326">
    <gml:pos>-67.563 -13.834</gml:pos>
  </gml:Point>
</gml:location>
```

Figure 4-41: Geographical 2D position in GML 3.1.0

EPSG 4326 is the two dimensional WGS-84 representation. If a position should be represented in three dimensions, that is with an altitude as well, then EPSG 4979 should be used (see Figure 4-42). Specifically the altitude is provided in meters above the geoid. The geoid approximates mean sea level (MSN). The shape of the geoid ellipsoid was calculated based on the hypothetical equipotential gravitational surface. A significant difference (up to 30m) however exists between this mathematical model and measurements taken with a GPS receiver, that measures altitude above theoretical sea level estimated by a World Geodetic System (WGS84) ellipsoid, which does not perfectly follow the theoretical geoid MSL.

```
<gml:position>
  <gml:Point srsName="urn:ogc:def:crs:EPSG:6.6:4979">
    <gml:pos>37.775 -122.422 22</gml:pos>
  </gml:Point>
</gml:position>
```

Figure 4-42: Geographical 3D position in GML 3.1.0

In order to express an area [Winterbottom] recommends a circular or polygon shape (see Figure 4-43 and Figure 4-44 respectively).

```
<?xml version="1.0"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf"
          xmlns:pidf="urn:ietf:params:xml:ns:pidf"
          xmlns:gp="urn:ietf:params:xml:ns:pidf:geopriv10"
          xmlns:gml="http://opengis.net/gml"
          entity="pres:user@example.com">
  <tuple id="a6fea09">
    <status>
      <gp:geopriv>
        <gp:location-info>
          <gml:extentOf>
            <gml:Polygon>
              <gml:exterior>
                <gml:Ring>
                  <gml:curveMember>
                    <gml:Curve>
                      <gml:segments>
                        <gml:CircleByCenterPoint >
                          <gml:pos
                            srsName="urn:EPSG:geographicCRS:4326">
                            42.5463 -73.2512
                          </gml:pos>
                          <gml:radius uom="urn:EPSG:uom:9001">
                            850.24
                          </gml:radius>
                        </gml:CircleByCenterPoint>
                      </gml:segments>
                    </gml:Curve>
                  </gml:curveMember>
                </gml:Ring>
              </gml:exterior>
            </gml:Polygon>
          </gml:extentOf>
        </gp:location-info>
        <gp:usage-rules>
        </gp:usage-rules>
      </gp:geopriv>
    </status>
    <timestamp>2004-12-01T09:28:43+10:00</timestamp>
  </tuple>
</presence>
```

Figure 4-43: PIDF document specifying a circular area

```
<?xml version="1.0"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf"
          xmlns:pidf="urn:ietf:params:xml:ns:pidf"
          xmlns:gp="urn:ietf:params:xml:ns:pidf:geopriv10"
          xmlns:gml="http://opengis.net/gml"
          entity="pres:user@example.com">
  <tuple id="a6fea09">
    <status>
      <gp:geopriv>
        <gp:location-info>
          <gml:extentOf>
            <gml:Polygon>
              <gml:exterior>
                <gml:LinearRing>
                  <gml:posList
                    srsName="urn:EPSG:geographicCRS:4326">
                    42.556844 -73.248157
                    42.549631 -73.237283
                    42.539087 -73.240328
                    42.535756 -73.254242
                    42.542969 -73.265115
                    42.553513 -73.262075
                    42.556844 -73.248157
                  </gml:posList>
                </gml:LinearRing>
              </gml:exterior>
            </gml:Polygon>
          </gml:extentOf>
        </gp:location-info>
        <gp:usage-rules>
        </gp:usage-rules>
      </gp:geopriv>
    </status>
    <timestamp>2004-12-13T14:49:53+10:00</timestamp>
  </tuple>
</presence>
```

Figure 4-44: PIDF document specifying a polygon area

4.5.6.3 Subscription Filters

A terminal-based location enabler makes applications possible, that are interested in triggered location notifications:

Tracing applications want to get a location update notification, every time the presentity has moved for a specified distance compared to the last update position.

Area watching applications are interested in notification, when a presentity enters or leaves a specified area.

Most network-based location enablers do not support triggered updates and applications thus have to fall back on polling schemes that do not scale at all. It is the most important advantage of a terminal-based location enabler that it supports triggered location updates in a simple and scalable way. This means that the trigger logic has to be implemented in the terminal so that necessary location updates are kept to an absolute minimum and scarce resources in the radio access network are used in the most economic way.

The trigger condition is specified by the watcher in the body of the SUBSCRIBE message, that gets sent to the presentity. Our research group has identified two mechanisms to express location triggers in the subscription:

- Event notification triggers [RFC 4660]
- Geopriv location filters [Mahy]

4.5.6.3.1 Event Notification Filters

Event notification filters ([RFC 4660], [RFC 4661]) can be used in the IMS to enable complex notifications such as setting up a notification when an entry in the buddy-list is at home and available for a video call. It is proposed to use event filtering to enable more complex location data notifications than pure location coordinates. The filter is sent within the SUBSCRIBE message from the watcher to the presence server.

The filter XML document specifies the content to be delivered to the watcher in the <what> element by including or excluding parts of a PIDF document with XPATH expressions. Furthermore the filter specifies when a notification shall be sent out in the <trigger> element. Actions can be triggered by elements being added, removed or changed in the current PIDF document, compared to the previously sent out document. Changes can be further specified by giving an original value of the changed element in a “from” attribute, or by giving a resulting value of the changed element in a “to” or by indicating a minimum change in the elements value in a “by” attribute.

Position update notifications can therefore by triggered by specifying a minimum change in position with the “by” attribute.

Definition of a trigger when a user enters a certain area can be specified with a deviation from a central point. The coordinates of the central point are given in the “to” attribute and the delta in the “by” attribute. Note however that the interpretation of the “to” attribute as latitude and longitude and the “by” value as a deviation from a given central point is not strictly specified in [RFC 4661].

Definition of a trigger when a user leaves a certain area is more difficult. Giving a center point in the “from” attribute and minimum distance change in thy “by” attribute leads to continuous notifications, once the user left the specified area.

For an area notification trigger, that fires only once when the user either enters or exits the specified area, it is proposed to give both the “from” and “to” attributes with the same position and a distance to that position in the “by” attribute.

By combining several trigger criteria with AND and OR more complex shapes can be created. This case is especially interesting since the ParlayX location web service API [ParalyX Loc] offers this kind of functionality to its client applications.

```
<?xml version="1.0" encoding="UTF-8"?>
<filter-set xmlns="urn:ietf:params:xml:ns:simple-filter">
  <ns-bindings>
    <ns-binding prefix="pidf" urn="urn:ietf:params:xml:ns:pidf"/>
    <ns-binding prefix="gp" urn="urn:ietf:params:xml:ns:pidf:geopriv10"/>
    <ns-binding prefix="gml"
            urn="urn:opengis:specification:gml:schema-xsd:feature:v2.0"/>
  </ns-bindings>
  <filter id="123" uri="sip:presentity@ftw.at">
    <what>
      <include>
        /pidf:presence/pidf:tuple/gp:geopriv/gp:location-info/gml:location
      </include>
    </what>
    <trigger> <!-- presentity went south or north -->
      <changed from="-67.563 -13.834" by="0.2 0.2">
        /pidf:presence/pidf:tuple/gp:geopriv/gp:location-nfo/gml:location/
          gml:point/gml:pos
      </changed>
    </trigger>
  </filter>
</filter-set>
```

Figure 4-45: Example area notification filter

The filter notation shown in Figure 4-45 has two draw-backs:

GML location formats:

GML supports a series of formats for coordinates. Since they rely on different XML constructs, the XML paths in the notification filters have to be adapted to the format used in the presence document. Alternatively, the author of the notification filter can set triggers for all possible formats and link them using the logical OR operation. More generally, a watcher can set a notification filter on any element in the presence document addressable by an XML path. This is important, if the watcher wants to subscribe to advanced features of the GML mark-up inside the presence document. The only restriction is that the watcher's filter needs to be compatible to the XML structure of the presence document. Since GML is a very powerful and flexible language this means the presence system will be either very complex or allow only a subset of the GML.

Accuracy:

A filter cannot be used to specify a required minimum accuracy of a numeric attribute such as location, since there is no <if> element in filters. The accuracy can be selected within the <what> element of the filter. The <what> element however is delivered to the watcher together with the location information in the NOTIFY message.

4.5.6.3.2 Geopriv location filters

Recently a draft for location filters has been proposed in [Mahy] to the IETF Geopriv working group. The <location-filter> element defines triggers, if a resource has moved a specified distance in horizontal or vertical direction, if a resource has exceeded a specified velocity, if a value (identified by an XPATH expression) in a PIDF document changes, or if the resource enters/exits a certain area.

It is proposed to use the location filter according to [Mahy] in a SIP event filter document according to [RFC 4661] (see Figure 4-46).

```
<?xml version="1.0" encoding="UTF-8"?>
 <filter-set xmlns="urn:ietf:params:xml:ns:simple-filter"
             xmlns:gml="urn:opengis:specification:gml:schema-
xsd:feature:v2.0"
             xmlns:my="urn:ftw:at">
   <ns-bindings>
     <ns-binding prefix="pidf" urn="urn:ietf:params:xml:ns:pidf"/>
     <ns-binding prefix="gp" urn="urn:ietf:params:xml:ns:pidf:geopriv10"/>
   </ns-bindings>
   <filter id="123" uri="sip:presentity@ftw.at">
     <what>
       <include>
         /pidf:presence/pidf:tuple/gp:geopriv/gp:location-info
       </include>
     </what>
     <location-filter>
        <enterOrExit>
          <my:Building gml:id="1jkdf9e54t7">
            <gml:name>FTW Main Building</gml:name>
            <gml:extentOf>
              <gml:Polygon>
                <gml:exterior>
                  <gml:LinearRing>
                    <gml:posList
             srsName="http://www.opengis.net/gml/srs/epsg.xml/#4979">
                          37.41188 -121.93243 0
                          37.41188 -121.93132 0
                          37.41142 -121.93132 0
                          37.41142 -121.93242 0
                          37.41188 -121.93243 0
                        </gml:posLis>
                      </gml:LinearRing>
                    </gml:exterior>
                  </gml:Polygon>
                </gml:extentOf>
              </my:Building>
            </enterOrExit>
     </location-filter>
   </filter>
 </filter-set>
```

Figure 4-46: Location filter document

The location filter uses a proprietary <Building> element, that is extended by GML 3.1. The <Building> element is uniquely identified by a gml:id attribute. The <Building> element contains a <gml:name> element and describes an area by a polygon. GML also specified the use of URLs in an xlink reference, so that the building boundary can be stored on a server (see Figure 4-47).

```
     <location-filter>
        <enterOrExit>
          <my:Building
            xlink:href=http://www.ftw.at/loc-defs/bldg/1jkdf9e54t7 />
          </my:Building>
        </enterOrExit>
     </location-filter>
```

Figure 4-47: Location Filter reference

[Mahy] also specifies a schema for describing a resource's location relative to a region or list of regions which might contain the resource. These regions can be defined dynamically in an <enterOrExit> element in a subscription filter. The <pidfResource> element is placed inside the location-info element in a PIDF-LO document. The <pidfResource> element can contain zero or more <containment> elements. Each containment element has a GML Feature sub-

element (of type "FeaturePropertyType") and a mandatory attribute which specifies if the PIDF resource is inside or outside of the feature, or if the position of the resource with respect to the region or region list is undefined (see Figure 4-48).

```
<?xml version="1.0" encoding="UTF-8"?>
  <presence xmlns="urn:ietf:params:xml:ns:pidf"
     xmlns:gp="urn:ietf:params:xml:ns:pidf:geopriv10"
     xmlns:pr="urn:ietf:params:xml:ns:pidf:geopriv10:containment"
     entity="pres:geotarget@example.com">
   <tuple id="sg89ae">
    <status>
     <gp:geopriv>
       <gp:location-info>
         <pr:pidfResource>
           <pr:containment position="inside">
            <my:Building>
               <gml:name>Building 10</gml:name>
             </my:Building>
           </pr:containment>
           <pr:containment position="outside">
             <my:ParkingGarage
xlink:href="http://server.example.com/loc-defs/bldg-mgr/parking"/>
           </pr:containment>
           <pr:containment position="undefined">
             <my:ConferenceRooms
xlink:href="http://server.example.com/loc-defs/userdef/confrooms"/>
           </pr:containment>
         </pr:pidfResource>
       </gp:location-info>
       <gp:usage-rules>
         <gp:retransmission-allowed>yes</gp:retransmission-allowed>
         <gp:retention-expiry>2003-06-23T04:57:29Z
         </gp:retention-expiry>
        </gp:usage-rules>
     </gp:geopriv>
    </status>
    <timestamp>2003-06-22T20:57:29Z</timestamp>
   </tuple>
  </presence>
```

Figure 4-48: PIDF document with containment information

4.5.7 Performance and Cost Benefits

Above analysis of the 3GPP Location Enabler architecture identified several shortcomings. First, complex request routing is required between GMLC and LIMS-IWF components in order to resolve the user's MSISDN and to accommodate roaming scenarios. In an IMS-enabled system this routing should be replaced by SIP message routing with its positive implications on performance, network management and security. It is therefore proposed to send a location request by using SIP routing procedures instead of introducing complex interworking functions.

Second the proposed network-based positioning methods are either insufficient regarding accuracy or require up-front investments in the core network. Accessing a terminal-based GPS receiver as a source for location data instead is a network technology independent solution that offers satisfactory accuracy while requiring only reasonable investments in those customers who want to use a location service.

Third, the privacy requirements and mechanisms proposed by 3GPP standardization for Location Services are similar to those of presence systems [RFC 4079] [3GPP TS 22.071]. Beyond the specified functions, a real world location service system requires components that allow provisioning, user access and user control of privacy parameters. It seems reasonable to re-use

the IMS presence infrastructure for subscription, authorization and privacy management of Location Based Services.
Fourth, a network-based positioning method is forced to implement triggered location updates through polling. In order to conserve battery power mobile terminals tend to be in an idle state most of the time. Hence position changes are not visible to the network. It is proposed to implement trigger logic in the terminal so that necessary location updates are kept to an absolute minimum and scarce resources in the radio access network are used in the most economic way.

4.5.8 Related Work

The Geographical Location and Privacy (GEOPRIV) working group of the IETF [Geopriv], [Mahy] has investigated a number of problems related to the distribution of geographical information on the Internet, from both a security and a policy angle. The result of this work is a generic framework for the creation and distribution of location information on the Internet that enables confidentiality and policy directives, which are abstracted from the format of the location information. Küpper and Treu propose complex location update strategies in the mobile terminal in order to realize scalable Location Based Services [Küpper]. Shaham et al. propose to use SIP event mechanism to transport location data [Shaham]. Polk and Rosen present a framework and requirements for usage of SIP to convey user location information and consider cases where message routing by intermediaries is influenced by the location of the session initiator [Polk]. In [Zundt] a general de-centralized location management is proposed that uses peer-to-peer technology for minimizing privacy concerns for location based services, however no connection to the already existing IMS presence framework is given. The work presented in [Mosmondor] uses the IMS presence framework for location services, but does not introduce a peer-to-peer architecture with a terminal-based location enabler.

5 Performance Analysis of a Location Presence System

This thesis proposes a terminal based location enabler for the IP Multimedia Subsystem. Subscriptions for location presence and notifications about changes of the watched user's position are transported in the network using the SIP event framework.

This chapter contains a detailed performance analysis for a carrier-grade deployment of a Location Service (LCS) based on the proposed presence location architecture. This chapter is organized as follows. After an introduction of the basic concepts of performance evaluation, a general model for a multi threaded location presence server is developed based on an M/M/c/k queuing model. Then availability considerations are presented based on a state model for the Mean Time To Failure (MMTF) and Mean Time To Repair (MMTR) times. The general Location Presence Server model and the availability model are combined and a method is presented to calculate the number of servers that have to run in a cluster in order to reach a given system availability and a given average system throughput.

Then a state model of a Location Presence Server is presented that implements a non preemptive priority queuing system with n queues having different priority. This model is used to calculate performance measures like the expected number of jobs in the system, the throughput of the system, the blocking probability for a job with the highest priority or the waiting probability for a job with lower priority.

Next the interarrival times of position update notifications are investigated by simulating user movement by a Random Waypoint model. It is shown that the proposed location presence architecture reduces the number of notifications substantially compared to a polling system.

Finally a fitting algorithm is investigated that is used to approximate simulated or measured distribution functions of interarrival times by models based on the phase concept.

5.1 Performance Evaluation Background

This section presents an overview of the basic concepts and terminology of performance evaluation that is used in this chapter. This section is an excerpt from [Allen], [Belch], [Kleinrock] and [Kreyszig] where a more comprehensive discussion on the topic can be found.

5.1.1 Kendall Notation

In queueing theory, Kendall's notation is the standard system used to describe and classify the queueing model that a queueing system corresponds to. First suggested by D. G. Kendall in 1953 as a 3 factor A/B/C notation system for characterising queues, it has since been extended to include up to 6 different factors.

A queue is described in shorthand notation by A/B/C/K/N/D or the more concise A/B/C. In this concise version, it is assumed K = ∞, N = ∞ and D = FIFO.

- A stands for the description of the arrival process (e.g., M is used for a Markovian (Poisson) arrival process, GI for a general independent arrival process).
- B stands for the service time distribution (e.g., M is used for a Markovian service with negative exponentially distributed service time, G for general distributed service times).
- C stands for the number of servers.
- K stands for the number of places in the system. The capacity of the system, or the maximum number of customers allowed in the system including those in service. When the number is at this maximum, further arrivals are turned away. If this number is omitted, the capacity is assumed to be unlimited, or infinite. Note: the number of places in the queue is K-C.
- N stands for the calling population. The size of the population from which the jobs come. If this number is omitted, the population is assumed to be unlimited.
- D stands for the queueing discipline (e.g., FIFO – First In First Out, LIFO - Last In First Out).

The simplest queue to analyze is the M/M/1 queue, where job interarrivals and service times are Markovian and there is a single server in the system.

5.1.2 Random Variables

A random variable is a function that describes the results of a random experiment.

A *discrete random variable* can only assume non-negative integer values. A discrete random variable X is described by the set of possible values and by the probabilities of each of these values. The set of probabilities is called the probability mass function (pmf). Thus, if the possible values of the random variable X are the non-negative integers, the pmf is given by the probabilities p_k

$p_k = P(X = k)$, for $k = 0,1,2\ldots,$

that is the probability that the random variable X assumes the value k.

The probabilities must meet the following requirements:

$P(X = k) \geq 0,$

$\sum_{all\ k} P(X = k) = 1$

Several important parameters can be derived from a discrete random variable.

Mean value or expected value:

$$E[X] = \sum_{all\,k} kP(X = k)$$

The function of a random variable is another random variable with the expected value:

$$E[f(X)] = \sum_{all\,k} f(k)P(X = k)$$

The *n*th moment is the expected value of the *n*th power of X (the 1st moment is simply the mean of X):

$$E[X^n] = \sum_{all\,k} k^n P(X = k)$$

The *n*th central moment is the expected value of the nth power of the difference between X and its mean:

$$E[(X - E[X])^n] = \sum_{all\,k} (k - E[X])^n P(X = k)$$

The second central moment is called the variance of X

$$\sigma_X^2 = \mathrm{var}(X) = E[(X - E[X])^2]$$

where σ_X is called the standard deviation.

The coefficient of variation is the normalized standard deviation:

$$c_X = \frac{\sigma_X}{E[X]}$$

A c*ontinuous random variable* X can assume all values in the interval [a,b], where

$$-\infty \le a < b \le +\infty$$

The probabilities for outcomes of a continuous random variables are described by its distribution function (also called cumulative distribution function – CDF):

$$F_X = P(X \le x)$$

The CDF describes the probability that the random variable X takes a value less or equal than x, for every x.

The CDF has the following properties for x<y:

$$F_X(x) \le F_X(y)$$

$$P(x < X \le y) = F_X(y) - F_X(x)$$

The probability density function (pdf) $f_X(x)$ can be used to describe the continuous random variable X, provided that $F_X(x)$ is differentiable:

$$f_X(x) = \frac{dF_X(x)}{dx}$$

Some properties of the pdf are:

$$f_X(x) \ge 0 \quad \text{for all x}$$

$$\int_{-\infty}^{\infty} f_X(x)dx = 1$$

$$P(x_1 \le X \le x_2) = \int_{x_1}^{x_2} f_X(x)dx$$

$$P(X = x) = \int_{x}^{x} f_X(x)dx = 0$$

$$P(X > x_3) = \int_{x_3}^{\infty} f_X(x)dx$$

The pdf of a continuous random variable is analogous to the pmf of a discrete random variable.
Several important parameters can be derived from a continuous random variable.
Mean or expected value:

$$E[X] = \int_{-\infty}^{\infty} x f_X(x)dx$$

Function of a radom variable:

$$E[g(X)] = \int_{-\infty}^{\infty} g(x) f_X(x)dx$$

nth moment:

$$E[X^n] = \int_{-\infty}^{\infty} x^n f_X(x)dx$$

nth central moment:

$$E[(X - E[X])^n] = \int_{-\infty}^{\infty} (x - E[X])^n f_X(x)dx$$

Variance, standard deviation and coefficient of variation are calculated in the same way as for a discrete random variable.
A random experiment may result in more than one random variable. The *joint probability mass function* of the discrete random variables $X_1, X_2,\ldots,X_n$ is given by:

$$P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n)$$

The *joint distribution function* for continuous random variables is given by:

$$F_X(x) = P(X_1 \le x_1, X_2 \le x_2, \ldots, X_n \le x_n)$$

The random variables $X_1, X_2,\ldots,X_n$ are said to be independent if

$$P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n) = P(X_1 = x_1) \cdot P(X_2 = x_2) \cdot \ldots \cdot P(X_n = x_n)$$

in the discrete case and

$$P(X_1 \le x_1, X_2 \le x_2, \ldots, X_n \le x_n) = P(X_1 \le x_1) \cdot P(X_2 \le x_2) \cdot \cdots \cdot P(X_n \le x_n)$$

in the continous case.
A conditional probability

$$P(X_1 = x_1 \mid X_2 = x_2, \ldots, X_n = x_n)$$

is the probability that $X_1=x_1$ under the condition that $X_2=x_2,\ldots, X_n=x_n$ and is calculated by

$$P(X_1 = x_1 \mid X_2 = x_2, \ldots, X_n = x_n) = \frac{P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n)}{P(X_2 = x_2, \ldots, X_n = x_n)}$$

For continuous random variables the conditional probability is:

$$P(X_1 \le x_1 \mid X_2 \le x_2, \ldots, X_n \le x_n) = \frac{P(X_1 \le x_1, X_2 \le x_2, \ldots, X_n \le x_n)}{P(X_2 \le x_2, \ldots, X_n \le x_n)}$$

The expected sum of (not necessarily independent) random variables is equal to the sum of the expected values of these random variables:

$$E\left[\sum_{i=1}^{n} c_i X_i\right] = \sum_{i=1}^{n} c_i E[X_i]$$

For independent random variables, the expected product of these random variables equals the product of the expected values:

$$E\left[\prod_{i=1}^{n} X_i\right] = \prod_{i=1}^{n} E[X_i]$$

The *covariance* of the random variables X and Y is a way to measure the dependency of X and Y

$$\operatorname{cov}[X,Y] = E[(X - E[X])(Y - E[Y])] = E[X \cdot Y] - E[X]E[Y]$$

If X and Y are statistically independent then

$$E[X \cdot Y] = E[X]E[Y]$$

$$\operatorname{cov}[X,Y] = 0$$

If X=Y, the covariance is

$$\operatorname{cov}[X,X] = E[X^2] - E[X]^2 = \operatorname{var}(X) = \sigma_X^2$$

The *correlation coefficient* is the covariance normalized to the product of standard deviations:

$$cor[X,Y] = \frac{\operatorname{cov}[X,Y]}{\sigma_X \sigma_Y}$$

If X=Y

$$cor[X,Y] = \frac{\sigma_X^2}{\sigma_X^2} = 1$$

5.1.3 Negative Exponential Distribution

The *negative exponential distribution* is the most important distribution in queueing theory. Random variables that have a negative exponential distribution are described by:

$$F_X(x) = \begin{cases} 1 - e^{-\lambda x}, & 0 \le x < \infty \\ 0 \end{cases}$$

$$E[X] = \frac{1}{\lambda}$$

$$f_X = \lambda e^{-\lambda x}$$

$$\operatorname{var}(X) = \frac{1}{\lambda^2}$$

$$c_X = 1$$

The negative exponential distribution is completely determined by its parameter λ that is the average rate of X.

The importance of the negative exponential distribution is based on the fact that it is the only continuous distribution that possesses the memoryless property:

$$P(X > s+t \mid X > s) = \frac{P(X > s+t,\, X > s)}{P(X > s)} = \frac{P(X > s+t)}{P(X > s)} = \frac{e^{-\lambda(s+t)}}{e^{-\lambda t}} = e^{-\lambda s}$$

This says that the conditional probability that one needs to wait more than another s seconds before the first arrival, given that the first arrival has not yet happened after t seconds, is no different from the initial probability that one needs to wait more than s seconds for the first arrival.

5.1.4 Stochastic and Markov Processes

A *stochastic process* is a family of random variables

$\{X_t : t \in T\},\ T \subseteq \mathbf{R}_+ = [0,\infty)$

where each random variable X_t is indexed by the parameter t (time).

The set of all possible values of X_t is called *state space* S of the stochastic process.

If the parameter set T is discrete and countable, the process is called a *discrete-parameter* process, otherwise it is called a *continuous-parameter* process.

If the state space S is discrete, the stochastic process is called a *chain*.

A stochastic process can be characterized by the joint cumulative distribution function

$$F_{X_1 X_2 \cdots X_n}(x_1, x_2, \ldots, x_n; t_1, t_2, \ldots, t_n) = F_{\mathbf{X}}(\mathbf{x};\mathbf{t}) = P[X(t_1) \le x_1, X(t_2) \le x_2, \ldots, X(t_n) \le x_n]$$

A stochastic process is called stationary, if the joint CDF is invariant to time shifts:

$$F_{\mathbf{X}}(\mathbf{x};\mathbf{t}+\tau) = F_{\mathbf{X}}(\mathbf{x};\mathbf{t})$$

The joint probability density function is defined by

$$f_{\mathbf{X}}(\mathbf{x};\mathbf{t}) = \frac{\partial F_{\mathbf{X}}(\mathbf{x};\mathbf{t})}{\partial \mathbf{x}}$$

A *Markov chain* is a process where the conditional CDF for $X_{t_{n+1}}$ depends only on the last previous value X_{t_n} and not on earlier values. This memoryless or Markov property may be written as:

$$P[X(t_{n+1}) = x_{n+1} \mid X(t_n) = x_n, X(t_{n-1}) = x_{n-1}, \ldots, X(t_1) = x_1] = P[X(t_{n+1}) = x_{n+1} \mid X(t_n) = x_n]$$

5.1.5 Discrete Time Markov Chains

A discrete time Markov chain (DTMC) is a Markov process with a discrete state space and a discrete parameter.

A system is said to be in state E_i at time n (or the *n*th step) for t = 1,2,…,n, when X_n=i. If the transition probabilities from state E_i to state E_j are independent of the parameter n, the Markov chain is called *homogeneous* and the (one-step) transition probabilities are defined as:

$$p_{ij} = P[X_n = j \mid X_{n-1} = i]$$

The m-step transition probability is defined as

$$p_{ij}^{(m)} = P[X_{n+m} = j \mid X_n = i]$$

From the Markov property is follows that

$$p_{ij}^{(m)} = \sum_k p_{ik}^{(m-1)} p_{kj} \quad m = 2,3,\ldots$$

This last equation is the *Chapman-Kolmogorov* equation for homogeneous Markov chains.

A Markov chain is called *irreducible*, if every state can be reached from every other state. This means there exists an integer m_0 such that

$$p_{ij}^{(m_0)} > 0$$

Let A be the set of all states in a Markov chain. A subset of states A_1 is called *closed* if there is no one-step transition possible from any state in A_1 to any state in A_1^c (the complement set of A_1). If A_1 consists of a single state, then this state, say E_i, is called an *absorbing* state and $p_{ii}=1$.

If A is closed and does not contain any proper subset that is closed, the corresponding Markov chain is irreducible. Otherwise the Markov chain is called *reducible*. If a closed subset of a reducible Markov chain is itself irreducible, this *irreducible sub-Markov chain* can be studied independently of the other states.

The probability that state E_j is visited again after n steps is defined by

$$f_j^{(n)} = P[\text{1st return to } E_j \text{ occurs n steps after leaving } E_j]$$

The probability of ever returning to E_j is given by

$$f_j = \sum_{n=1}^{\infty} f_i^{(n)} = P[\text{ever returning to } E_j]$$

E_j is called *recurrent* for $f_j = 1$.

E_j is called transient for $f_j < 1$.

If the number of steps between two visits of state E_j is constant, this state is called *periodic*, otherwise *aperiodic*.

For recurrent states the *mean recurrence time* is defined as

$$M_j = \sum_{n=1}^{\infty} n f_j^{(n)}$$

If $M_j = \infty$, then E_j is said to be *recurrent null*, otherwise *recurrent nonnull*.

The states of an irreducible Markov chain are either all transient or all recurrent nonnull or all recurrent null. If periodic, all states have the same period.

The probability of finding a Markov chain in state E_j at the *n*th-step shall be defined as:

$$\pi_j^{(n)} = P[X_n = j]$$

In an irreducible and aperiodic homogenous Markov chain the limiting probabilities π_j always exist and are independent of the initial state probability distribution:

$$\pi_j = \lim_{n\to\infty} \pi_j^{(n)}$$

Then either

a) all states are transient or all states are recurrent null in which case $\pi_j = 0$ for all j and there exists on stationary distribution
or

b) all states are recurrent nonnull and then $\pi_j > 0$ for all j, in which case the set $\{ \pi_j \}$ is a stationary probability distribution and

$$\pi_j = \frac{1}{M_j}$$

In this case the quantities π_j are uniquely determined through the following equations:

$$\sum_i \pi_i = 1$$

$$\pi_j = \sum_i \pi_i p_{ij}$$

A state E_j is said to be *ergodic*, if it is aperiodic, recurrent and not null. If all states of a Markov chain are ergodic, then the chain itself is said to be ergodic.

If the limiting stationary probability distributions $\{\pi_j\}$ converge independent of the initial state distribution, a Markov chain is ergodic. The limiting probabilities $\{\pi_j\}$ of an ergodic Markov chain are often referred to as the *equilibrium* probabilities.
Furthermore all states of a finite (having a finite number of states) aperiodic irreducible Markov chain are ergodic.
The *transition probability matrix* **P** is:

$$\mathbf{P} = [p_{ij}]$$

With the state probability vector

$$\boldsymbol{\pi} = [\pi_0, \pi_1, \ldots]$$

the system in equilibrium can be described by:

$$\boldsymbol{\pi} = \boldsymbol{\pi}\mathbf{P}$$

Graphically, a finite-state DTMC is represented by a state transition diagram, a finite directed graph, where state i of the chain is depicted by a vertex, and a one-step transition from state i to state j by an edge marked with one-step transition probability p_{ij}.

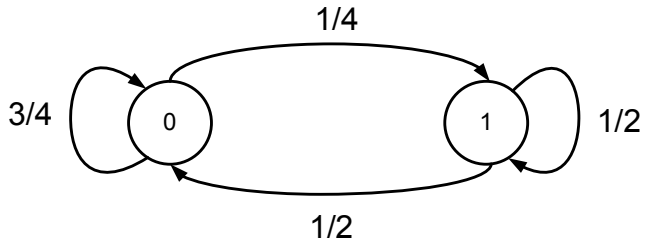


Figure 5-1: Example of state transition diagram

In Figure 5-1 an example of a state transition diagram is given. The corresponding state transition pobablility matrix is:

$$\mathbf{P} = \begin{pmatrix} \frac{3}{4} & \frac{1}{4} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}$$

Note that there will always be linear dependence in the equations of the equilibrium state probabilities that describe the system. In order to solve the system it is required that:

$$\sum_i \pi_i = 1$$

The description of the transient behaviour of the system involves the probability of state E_j at time n:

$$\boldsymbol{\pi}^{(n)} = [\pi_0^{(n)}, \pi_1^{(n)}, \pi_2^{(n)} \ldots]$$

This can be done iteratively by:

$$\boldsymbol{\pi}^{(1)} = \boldsymbol{\pi}^{(0)}\mathbf{P}$$

$$\boldsymbol{\pi}^{(2)} = \boldsymbol{\pi}^{(1)}\mathbf{P} = [\boldsymbol{\pi}^{(0)}\mathbf{P}]\mathbf{P} = \boldsymbol{\pi}^{(0)}\mathbf{P}^2$$

$$\boldsymbol{\pi}^{(n)} = \boldsymbol{\pi}^{(0)}\mathbf{P}^n$$

This iteration converges against the equilibrium state probabilities. The rapidity of the convergence depends on the eigenvalues of **P**.
The memoryless property of a Markov chain influences the time that the system spends in a given state E_i. With probability p_{ii}, the system remains in state E_i and with probability $(1-p_{ii})$ it leaves this state. The probability the the system stays in state E_i for exactly m steps is given by:
P[system remains in E_i for exactly m additional steps] = $(1-p_{ii})p_{ii}$

This is the fomula of a geometric distribution that is the only discrete distribution with memoryless property.

5.1.6 Continuous-Time Markov Chains

Continuous-time Markov chains allow the time parameter to assume any point in time.
The time dependent transition probability for time contiunous Markov chains is defined as

$$p_{ij}(s,t) = P[X(t) = j \mid X(s) = i]$$

Where X(t) is the state of the system at time t > s.
For three successive points in time t > u >s the Chapman-Kolmogorov equation can be written as:

$$p_{ij}(s,t) = \sum_k p_{ik}(s,u)\, p_{ki}(u,t) \quad \text{i,j=0,1,2…}$$

In matrix form the Chapman-Kolmogorov equation becomes:

$$\mathbf{H}(s,t) = [p_{ij}(s,t)]$$

$$\mathbf{H}(s,t) = \mathbf{H}(s,u)\mathbf{H}(u,t)$$

For the continuous-time Markov chain the time interval from s to t is now replaced by an infinitesimal Δt. This gives the forward Chapman-Kolmogrov equation:

$$\mathbf{P}(t) = [p_{i,j}(t,t+\Delta t)]$$

$$\mathbf{H}(s,t+\Delta t) = \mathbf{H}(s,t)\mathbf{H}(t,t+\Delta t)$$

$$\mathbf{H}(s,t+\Delta t) = \mathbf{H}(s,t)\mathbf{P}(t)$$

$$\mathbf{H}(s,t+\Delta t) - \mathbf{H}(s,t) = \mathbf{H}(s,t)\mathbf{P}(t) - \mathbf{H}(s,t) = \mathbf{H}(s,t)[\mathbf{P}(t) - \mathbf{I}]$$

$$\frac{\mathbf{H}(s,t+\Delta t) - \mathbf{H}(s,t)}{\Delta t} = \frac{\mathbf{H}(s,t)[\mathbf{P}(t) - \mathbf{I}]}{\Delta t}$$

$$\Delta t \to 0 \quad \frac{\partial \mathbf{H}(s,t)}{\partial t} = \mathbf{H}(s,t)\mathbf{Q}(t) \quad s \le t$$

$$\mathbf{Q}(t) = \lim_{\Delta t \to 0} \frac{\mathbf{P}(t) - \mathbf{I}}{\Delta t}$$

The matrix **Q**(t) is known as the *infinitesimal generator* matrix of the transition matrix function **H**(s,t). Another more descriptive name for **Q**(t) is the *transition rate* matrix. The elements of **Q**(t) are:

$$\mathbf{Q}(t) = [q_{ij}(t)]$$

$$q_{ii}(t) = \lim_{\Delta t \to 0} \frac{p_{ii}(t,t+\Delta t) - 1}{\Delta t}$$

$$q_{ij}(t) = \lim_{\Delta t \to 0} \frac{p_{ij}(t,t+\Delta t)}{\Delta t} \quad i \neq j$$

If the system is in state E_i at time t, then $-q_{ii}(t)$ is the rate at which the process departs from state E_i. Similarly, given that the system is in state E_i at time t, then $q_{ij}(t)$ gives the rate at which the process moves from E_i to E_j.
Furthermore the probability normation condition requires that

$$\sum_j p_{ij}(s,t) = 1$$

The probability normation condition implies that

$$\sum_j q_{ij}(t) = 0 \quad \text{for all i}$$

This means that rate at which the process departs from a state is equal to the sum of rates where the process arrives at this state.
In a similar fashion, the backward Chapman-Kolmogorov equation can be derived:

$$\frac{\partial \mathbf{H}(s,t)}{\partial t} = -\mathbf{Q}(t)\mathbf{H}(s,t) \quad s \le t$$

From the memoryless property of the Markov chain it follows that the state times are characterized by a negative exponential distribution.
If the continuous-time Markov chain is homogeneous, the time dependence can be dropped:

$$p_{ij}(t) = p_{ij}(s, s+t)$$

$$q_{ij} = q_{ij}(t) \quad i, j = 1,2,\ldots$$

$$\mathbf{H}(t) = \mathbf{H}(s, s+t) = [p_{ij}(t)]$$

$$\mathbf{Q} = \mathbf{Q}(t) = [q_{ij}]$$

The Chapman-Kolmogorv equation becomes:

$$p_{ij}(s+t) = \sum_k p_{ik}(s)\, p_{kj}(t)$$

$$\mathbf{H}(s+t) = \mathbf{H}(s)\mathbf{H}(t)$$

The forward and backward equations become, respectively,

$$\frac{dp_{ij}(t)}{dt} = q_{jj}\, p_{ij}(t) + \sum_{k \ne j} q_{kj}\, p_{ik}(t)$$

$$\frac{d\mathbf{H}(t)}{dt} = \mathbf{H}(t)\mathbf{Q}$$

and

$$-\frac{dp_{ij}(t)}{dt} = -q_{ii}\, p_{ij}(t) + \sum_{k \ne i} q_{ik}\, p_{kj}(t)$$

$$\frac{d\mathbf{H}(t)}{dt} = \mathbf{Q}\mathbf{H}(t)$$

A solution can be found with the common initial condition

$$\mathbf{H}(0) = \mathbf{I}$$

$$\mathbf{H}(t) = e^{\mathbf{Q}t}$$

For the state probabilities this leads to the following differential equations:

$$\frac{d\pi_j(t)}{dt} = q_{ij}\pi_j(t) + \sum_{k \ne j} q_{kj}\pi_k(t)$$

$$\frac{d\boldsymbol{\pi}(t)}{dt} = \boldsymbol{\pi}(t)\mathbf{Q}$$

For an irreducible homogeneous Markov chain it can be shown that the following limits always exist and are independent of the initial state of the chain, namely,

$$\lim_{t\to\infty} p_{ij}(t) = \pi_j$$

The set $\{\pi_j\}$ will from the limiting state probability distribution.
For an ergodic Markov chain there will be a further limit, which will be independent of the initial distribution, namely

$$\lim_{t\to\infty} \pi_j(t) = \pi_j$$

This limiting distribution is given uniquely as the solution of the following system of linear equations:

$$q_{jj}\pi_j + \sum_{k \neq j} q_{kj}\pi_k = 0$$

$$\boldsymbol{\pi}\mathbf{Q} = 0$$

This last equation coupled with the probability normation condition

$$\sum_j \pi_j = 1$$

gives the limiting state probabilities.

5.1.7 Steady-State Solutions of Markov Chains

This section discusses the solution of the equations of ergodic Markov chains in order to get the equilibrium state probabilities.

The matrix equations for DTMC

$$\boldsymbol{\pi} = \boldsymbol{\pi}\mathbf{P}$$

$$0 = \boldsymbol{\pi}(\mathbf{P} - \mathbf{I})$$

and for CTMC

$$0 = \boldsymbol{\pi}\mathbf{Q}$$

have both the form of a system of linear equations

$$0 = \mathbf{xA}$$

Due to the type of entries representing the parameters of a Markov chain, matrix **A** is singular and it can be shown that Matrix **A** is of rank n-1 for a system with n states. It follows immediately that the resulting system of linear equations is not linearly independent and that one of the equations is redundant.

To get a unique solution, a normalization condition must be imposed. One way to approach the solution is to directly incorporate the normalization condition

$$\mathbf{x1} = 1$$

This can be regarded as a substitution of one of the columns of matrix **A** by the unit vector $\mathbf{1} = [1,1,\ldots,1]^T$, getting such the matrix $\mathbf{A}^*$.

The resulting system of linear equations is

$$\mathbf{b} = \mathbf{xA}^* \quad \mathbf{b} = [0,0,\ldots,0,1]$$

To determine the steady-state probabilities of finite Markov chains, three different approaches for the solution of a linear system of the form $\mathbf{b} = \mathbf{xA}^*$ are commonly used:

- Direct numerical methods
- Iterative numerical methods
- Techniques that yield closed-form results.

Both types of numerical methods have merits of their own. Whereas direct methods yield exact results, iterative methods are generally more efficient, both in time and space. Disadvantages of iterative methods are that for some of these methods no guarantee of convergence can be given in general and that determination of suitable error bounds for termination of the iterations is not always easy. Since iterative methods are considerably more efficient in solving Markov chains, they are commonly used for larger models. For smaller models consisting of up to a few thousand states, direct methods are reliable and accurate. Though closed-form results are highly desirable, they can be obtained for only a small class of models that have some structure in their matrix.

In this thesis, direct numerical methods will be used to solve the linear equations for homogeneous Markov chains. Direct methods operate and modify the parameter matrix. They use a fixed amount of computation time independent of the parameter values and there is no issue of convergence. But they are subject to fill-in of matrix entries, that is, original zero entries can become non-zeros. This makes the use of sparse storage difficult. Direct methods are also subject to the accumulation of round-off errors.
Our performance evaluation research group at the Institute of Broadband Communication at the Vienna University of Technology has also implemented iterative numerical methods for the solution of CTMC systems [Sommereder], where Runge-Kutta methods with adaptive step size control were used to calculate the transient state probabilities and the flow times.
Among the techniques most commonly applied in direct numerical methods is the well known Gaussian elimination algorithm. A modification of the Gauss elimination is the LU-factorization [Kreyszig]:

$$\mathbf{M} = \mathbf{LU}$$

The essential part of Gaussian elimination is provided by the factorization of the parameter matrix **M** into the components of an upper triangular matrix **U** and a lower triangular matrix **L**. The idea is that the **L** and **U** can be computed directly (without using the Gauss elimination). The LU-factorization needs about $n^3/3$ operations, about half as many as the Gauss elimination.
Once matrix **M** has been factorized in to **L** and **U**, the solution of the system can be found with

$$\mathbf{Mx} = \mathbf{LUx} = \mathbf{b}$$

$$\mathbf{Ly} = \mathbf{b}$$

$$\mathbf{Ux} = \mathbf{y}$$

Where first **y** is calculated from **L** and **b** and then **x** is calculated from **U** and **y**.
Note, well that the matrix $\mathbf{A}^*$ for the steady state probabilities has to be transposed in order to fit the general form of the LU factorization algorithm:

$$\mathbf{Mx} = \mathbf{LUx} = \mathbf{b}$$

$$\mathbf{xA}^* = \mathbf{b}$$

$$(\mathbf{xA}^*)^T = \mathbf{b}$$

$$(\mathbf{A}^*)^T \mathbf{x}^T = \mathbf{b}$$

$$\mathbf{M} = (\mathbf{A}^*)^T$$

A system of linear equations is called *well-conditioned*, if a small error in the coefficients or in the solving process has only a small effect on the solution
A system of linear equations is called *ill-conditioned*, if a small error in the coefficients or in the solving process has a large effect on the solution. E.g., the solution of a system that represents two intersecting lines depends on the angle of intersection. If the angle between the lines is small, a small change in the coefficients leads to a large displacement of point of intersection.
Some symptoms of ill-conditioning of a system **Ax = b** are:

- $|\det \mathbf{A}|$ is small compared with the size of $\max(|a_{ij}|)$ and the terms in **b.**
- The entries in $\mathbf{A}^{-1}$ are large in absolute values compared with the absolute values of the solution.

A system is well-conditioned if:

- The main diagonal entries are large in absolute value compared with the absolute values of the other entries.

The residual **r** of an approximate solution $\tilde{\mathbf{x}}$ of **Ax = b** is

$$\mathbf{r} = \mathbf{b} - \mathbf{A}\tilde{\mathbf{x}}$$

$$\mathbf{A}\tilde{\mathbf{x}} = \mathbf{b} - \mathbf{r} = \mathbf{A}\mathbf{x} - \mathbf{r}$$

$$\mathbf{A}(\mathbf{x} - \tilde{\mathbf{x}}) = \mathbf{r}$$

In order to investigate the condition of a matrix a *matrix norm* is used. A matrix norm is derived from the definition of a *vector norm*.

The norm of a vector **x** in a vector space V is defined as:

$$\|\mathbf{x}\| : V \to \Re$$

a) $\|\mathbf{x}\|$ is a nonnegative real number

b) $\|\mathbf{x}\| = 0$ if an only if $\mathbf{x} = \mathbf{0}$

c) $\|k\mathbf{x}\| = |k|\|\mathbf{x}\|$ for all k

d) $\|\mathbf{x} + \mathbf{y}\| \le \|\mathbf{x}\| + \|\mathbf{y}\|$ (triangle inequality)

The vector norm can be interpreted as a generalized length.

The most frequently used vector norms are:

"l_1-norm": $\|\mathbf{x}\|_1 = |x_1| + \cdots + |x_n|$

"l2-norm, Euclidean norm": $\|\mathbf{x}\|_2 = \sqrt{x_1^2 + \cdots + x_n^2}$

"l_∞-norm": $\|\mathbf{x}\|_\infty = \max_j |x_j|$

The matrix norm ||**A**|| is defined as:

a) $\|\mathbf{A}\|$ is a nonnegative real number

b) $\|\mathbf{A}\| = 0$ if an only if $\mathbf{A} = \mathbf{0}$

c) $\|k\mathbf{A}\| = |k|\|\mathbf{A}\|$ for all k

d) $\|\mathbf{A} + \mathbf{B}\| \le \|\mathbf{A}\| + \|\mathbf{B}\|$ (triangle inequality)

e) $\|\mathbf{A} \cdot \mathbf{B}\| = \|\mathbf{A}\| \cdot \|\mathbf{B}\|$

For a given vector norm, the matrix norm of an *n* x *n* matrix is:

$$\|\mathbf{A}\| = \max \frac{\|\mathbf{A}\mathbf{x}\|}{\|\mathbf{x}\|}$$

From the l_1 vector norm follows the

"column sum norm": $\|\mathbf{A}\|_1 = \max_k \sum_{j=1}^{n} |a_{jk}|$

From the l_∞-norm follows the

"row sum norm": $\|\mathbf{A}\|_\infty = \max_j \sum_{k=1}^{n} |a_{jk}|$

The condition number κ(**A**) of a matrix **A** is defined as

$$\kappa(\mathbf{A}) = \|\mathbf{A}\|\|\mathbf{A}^{-1}\|$$

A system of linear equations **Ax=b** whose condition matrix is small is well-conditioned. A large condition number indicates ill-conditioning.

$$\frac{\|\mathbf{x}-\tilde{\mathbf{x}}\|}{\|\mathbf{x}\|}=\frac{\|\mathbf{A}^{-1}\mathbf{r}\|}{\|\mathbf{x}\|}\le\|\mathbf{A}^{-1}\|\|\mathbf{r}\|\frac{\|\mathbf{A}\|}{\|\mathbf{b}\|}=\kappa(\mathbf{A})\frac{\|\mathbf{r}\|}{\|\mathbf{b}\|}$$

If the condition number is small, a small $\frac{\|\mathbf{r}\|}{\|\mathbf{b}\|}$ indicates as mall relative error $\frac{\|\mathbf{x}-\tilde{\mathbf{x}}\|}{\|\mathbf{x}\|}$, so that the system is well conditioned. This does not hold for a large condition number.
In practice, the inverse Matrix $\mathbf{A}^{-1}$ will not be known, so that for computing the condition number $\kappa(\mathbf{A})$, the matrix norm $\|\mathbf{A}^{-1}\|$ must be estimated (e.g., see [Cline]).

5.2 Performance Evaluation of an IMS Location Presence Server

This section presents an analytical model of an IMS Location Presence Application Server. This model shall be used to perform workload analysis and availability considerations for a carrier-grade Location Presence Service installation.

Figure 5-2 shows the system architecture of an IMS Location Presence service. The source of location presence data is a mobile terminal, that is connected via a Bluetooth Personal Area Network (PAN) with a GPS (Global Positioning System) receiver. The terminal uses the IMS access network and the IMS core network (P-CSCF, I-CSCF, S-CSCF, HSS) to send location presence notifications to a Presence Application Server (AS) that is dedicated to the processing of location presence data. The Presence AS is a SIP AS in accordance with the IMS specifications, that is connected via the ISC protocol to the Serving-Call State Control Function (S-CSCF). Initial Filter Criteria (IFC) have to be provisioned for a presentity in the Home Subscriber Server (HSS), that trigger the processing of location presence subscriptions (SIP SUBSCRIBE message) and notifications (SIP NOTIFY message) by the Presence AS.

For more detailed information about the components of an IMS system, please refer to the chapter “IP Multimedia Subsystem – IMS”.

The Gateway Mobile Location Centre (GML) is shown in the architecture as an alternative source of location data. The GMLC offers a network based method for terminal positioning, that may be used as a fall-back, if the terminal of the presentity is not equipped with a GPS receiver, or GPS data is not available. Network-based location methods however do not offer the same accuracy as satellite positioning systems.

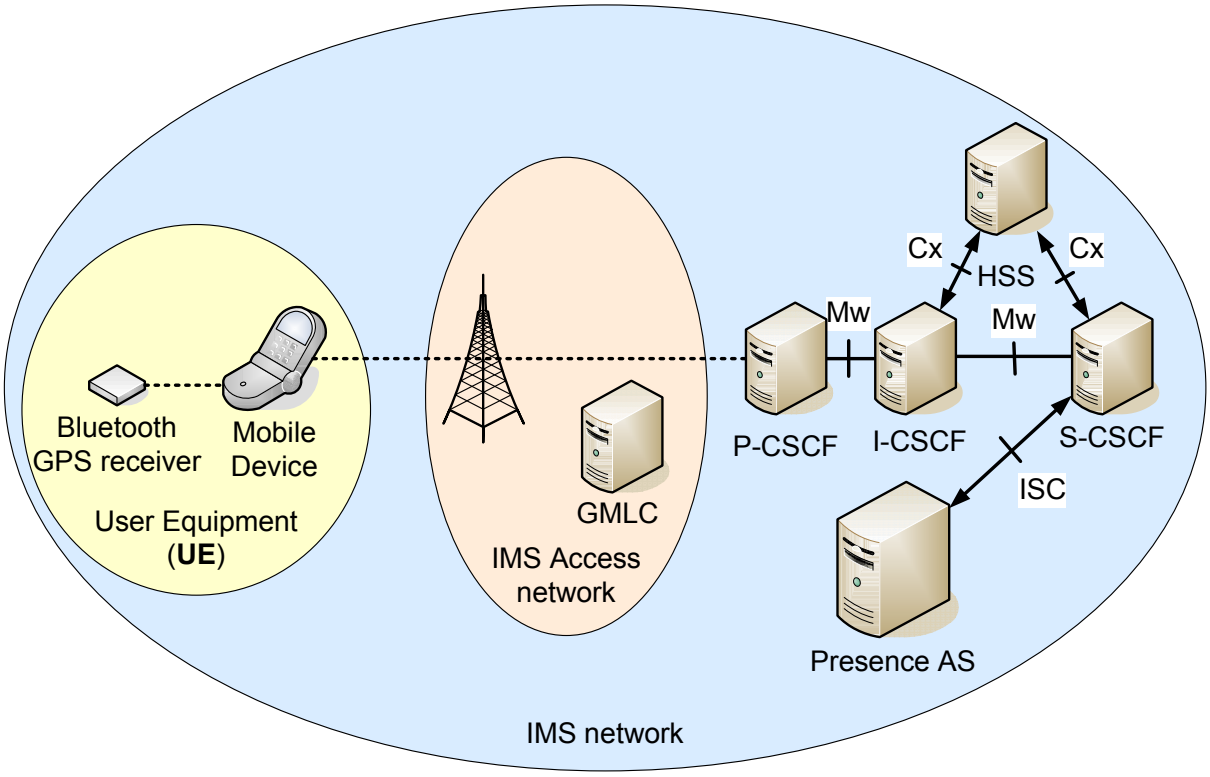


Figure 5-2: System architecture of an IMS Location Presence service

Figure 5-3 shows the protocol stacks of an IMS Location Presence system. The SIP Edge Presence Server in the terminal is responsible for comparing position data from the GPS re-

ceiver with filter criteria from the location presence subscription. JSR 179 is a Java API that enables generic access to positioning methods (e.g., GPS, Galileo, DHCP). If a filter matches, a notification is sent to the Presence AS. The figure shows also the Location Enabler as an intermediate node. The Location Enabler acts as a presence aggregator, that controls whether data from a terminal-based or a network-based positioning method is used. The SIP protocol is used for communication between the user's terminal and the Presence AS. JSR 180 is a SIP API for mobile devices.

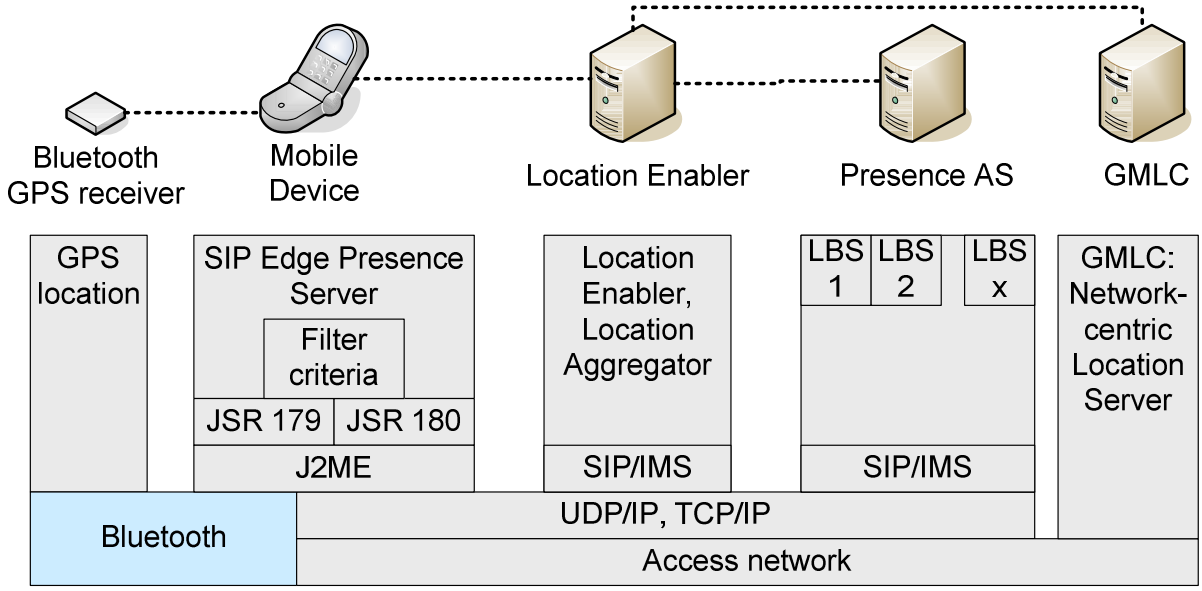


Figure 5-3: Protocol stacks of an IMS Location Presence system

Location Based Service (LBS) applications shall be classified regarding the required traffic intensity:

- Location 'Push/Pull' type applications, where the position of a terminal is requested once.
  An example of such a service is a weather forecast application that first locates the terminal and then offers a localized weather forecast.
- Location 'Tracking' applications, where the position of a terminal is requested on a periodical basis.
  Typical tracking applications are navigation systems, that need to keep track continuously of the user's position.

Figure 5-4 shows the message sequence diagram of a 'Push/Pull' type location application. In this example the presentity requests its own location. The request is sent to an application server that runs the 'Push/Pull' application. The application server forwards a subscription to the Presence AS. The Presence AS authenticates and authorizes the application server and send the subscription to the presentity. The Edge Presence Server on the presentity's terminal gets the current position from the GPS receiver and sends a notification to the Presence AS. The Presence AS forwards the NOTIFY message to the application server.
A 'Pull' application requires the user's location immediately after the subscription in a synchronous transaction, whereas a 'Push' application waits for asynchronous notifications.

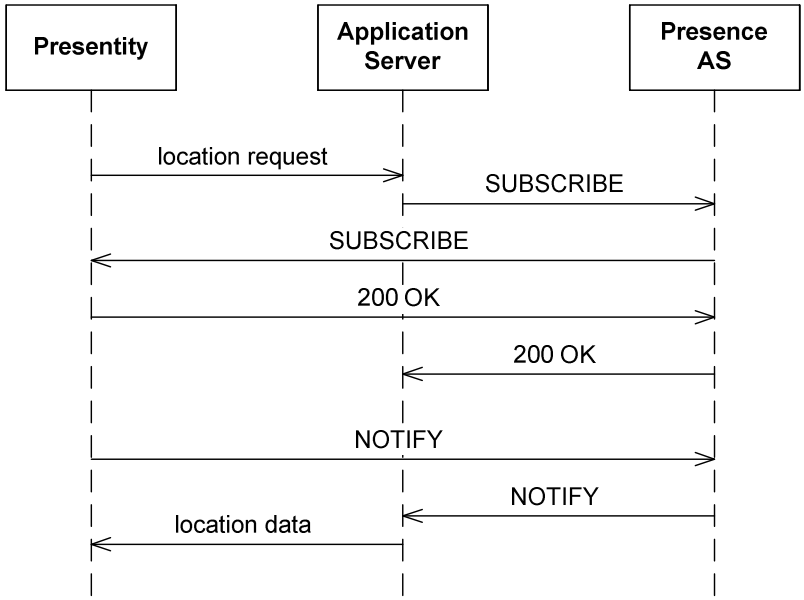


Figure 5-4: Message sequence diagram of a 'Push/Pull' type location application

Figure 5-5 shows the message sequence diagram of a 'Tracking' type location application. Here the watcher and the presentity are shown as different entities. The watcher sends the location request to an application server that runs the 'Tracking' application. The application server forwards a subscription to the Presence AS. The Presence AS authenticates and authorizes the application server and send the subscription to the presentity. The Edge Presence Server on the presentity's terminal now periodically gets the position from the GPS receiver and sends notifications to the Presence AS. The Presence AS forwards the NOTIFY messages to the application server.

The main difference to the 'Push/Pull' type application is, that one subscription may result in many notifications.

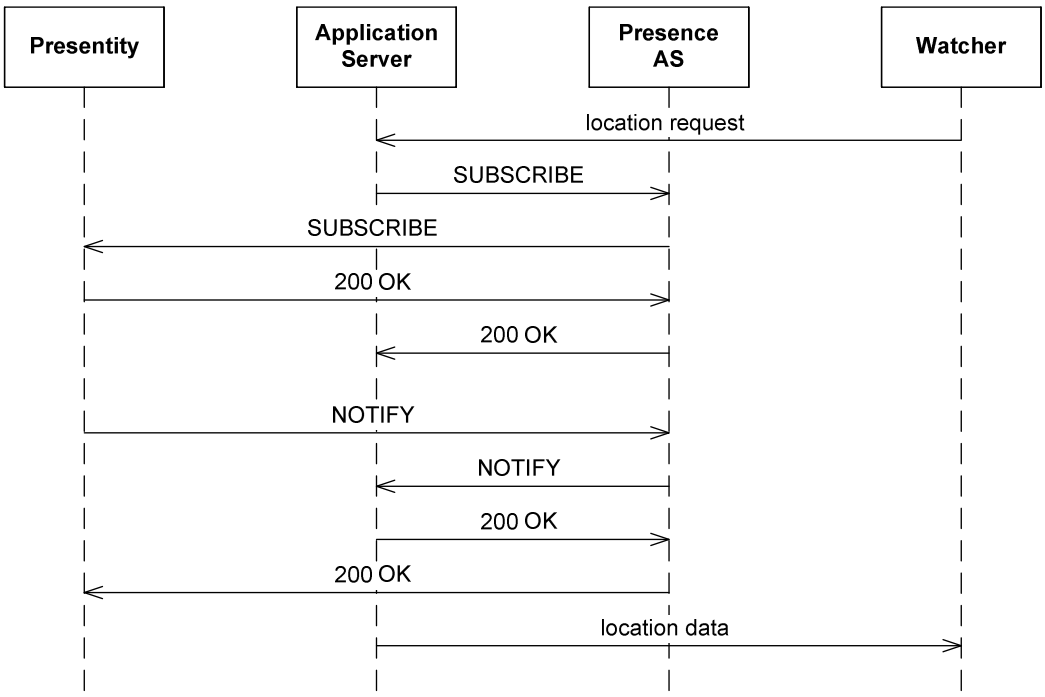


Figure 5-5: Message sequence diagram of a 'Tracking' type location application

5.2.1 Analytical Model of a Location Presence Application Server

The messages to and from an IMS Location Presence Application Server are characterized by the following considerations:

- The Presence AS acts as a SIP Back-to-Back User Agent (B2BUA). From a SIP point of view, this means that the SIP dialogues between the application and the Presence AS are decoupled from the SIP dialogues between the Presence AS and the Edge Presence Server in the user's terminal. Decoupling the dialogues has several reasons. First the authentication and authorization context between a potentially external application and the Presence AS is different than between the Presence AS and the user's terminal. Second subscription expiry times may be different. Third the Presence AS may aggregate data from another location enabler (e.g., a GMLC) into the notifications towards the application.
- The Presence AS creates an application SIP dialogue triggered by SUBSCRIBE message from the application server.
- During the application dialogue a user dialogue is started by the Presence AS where one down-link SUBSCRIBE transaction is executed.
- One up-link NOTIFY transaction towards the Presence AS is triggered by the subscription but started asynchronously by the user's terminal.
- The content of the NOTIFY message is forwarded by the presence server to the application server.

The first step to an analytical model of the presence server is to develop a model for subscriptions from the application server. The location presence server has to execute the following tasks on receiving a SUBCRIBE message from the application server:

- authentication and authorization of the request. This business logic usually involves access to a database system.
- triggering a subscription to the edge presence server on the user's terminal
- waiting for a 200 OK message that acknowledges the subscription.
- sending a 200 OK message towards the application server.

The interarrival time of SUBSCRIBE messages at the presence server shall be modelled by the random variable A_S that is characterized by the probability distribution function A_S (t). A_S (t) gives the probability that the time between two arrivals is less or equal the time t:

$$A_S(t) = P\{A_S \leq t\}$$

It is assumed that the arrivals are independent and identically distributed (iid) and thus represent a renewal process.

It is further assumed that the random variable A_S can be characterized by a negative exponential distribution function with the parameter α that corresponds to the mean arrival rate:

$$A_S(t) = 1 - e^{-\alpha t}$$

A renewal process with negative exponentially distributed interarrival times is called Poisson process, that has the Markov property of memorylessness. A stochastic process has the Markov property if the conditional probability distribution of future states of the process, given the present state and all past states, depends only upon the present state and not on any past states.

The second step to an analytical model of the presence server is to develop a model for notifications from the terminal. The location presence server has to execute the following tasks on receiving a NOTIFY message from the terminal:

- Loading of the application server subscription dialogue. For long running subscriptions the dialogue may have persisted into a database server
- Checking privacy and policy settings against a database system
- Sending a NOTIFY message to the application server
- Waiting for a 200 OK for the NOTIFY from the application server
- Acknowledging the NOTIFY from the terminal by sending a 200OK.

Notifications are triggered asynchronously by the terminal. One SUBSCRIBE message may trigger several NOTIFY messages. The ratio r of notifications per subscription is application dependent. Subscriptions have to be renewed before an expiry time specified by the application context.

The notification to subscription rate r gives the total SIP request rate of

$$\lambda = \alpha + r\alpha = (1+r)\alpha$$

at the location presence server.

It is further assumed that the arrivals of SIP requests at the location presence server are independent and identically distributed. The random variable of the SIP request interarrival time A_R shall be characterized by the probability distribution function $A_R(t)$.

$$A_R(t) = P\{A_R \le t\} = 1 - e^{-\lambda t} = 1 - e^{-(1+r)\alpha t}$$

Thus the arrival process of SIP requests at the location presence server is a Poisson process with a mean arrival rate of λ.

The third step to an analytical model of the presence server is to develop a model for the service process at the location presence server.

It is assumed that message transfer times in the radio access network and processing times at the terminal are at least an order of magnitude larger than message processing times at the presence location server and inter-server message transfer times. The location presence service process is therefore dominated by the communication process between presence server and terminal. Subscription and notification handling are similar from a processing point of view, as they require one request and one response each.

The time S between two requests served is modelled as a random variable with the probability distribution S(t). It is assumed that S(t) has a negative exponential probability distribution with the mean service rate μ.

The resulting model of a location presence server is further developed based on the following assumptions:

- Performance of SIP communication servers is mainly influenced by Input/Output processes on the network interfaces.
- Dimensioning of communication servers is therefore dominated by the number of parallel network ports that the server can serve in parallel.
- CPU and disk utilization are of secondary importance.

- One incoming request usually results in more than one outgoing network connections (fan-out) towards e.g., terminal, application server, database systems, business support systems
- The number of concurrent requests that can be served is limited by the maximum number of parallel processing threads in the server.
- Each SIP request is processed in an own processing thread.

These considerations lead to a location presence server model that is shown in Figure 5-6.

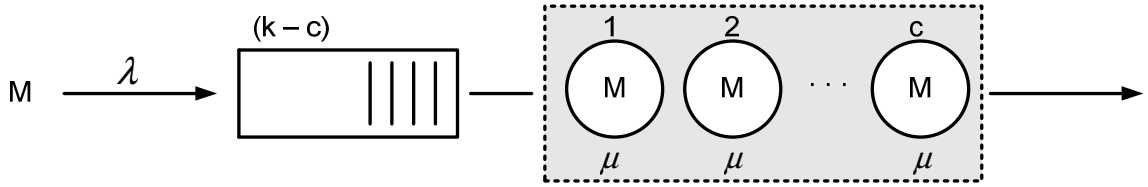


Figure 5-6: Location Presence server model

The location presence server is modelled as an M/M/c/k queuing system. The SIP request arrival process has Markov property with a mean arrival rate of λ. The system has c parallel server units (processing threads) that can each handle an arriving SIP requests. The service process has also Markov property with a mean service rate of μ. The location presence server system has a limited number k of requests that it can hold at a time. Requests that are not served are queued in a queue with k-c places. The system is blocked, if all server units are busy and the queue is full. Requests that arrive at a blocked system cannot be processed.
The M/M/c/k queuing system is described by the following formulas [Allen]:
The traffic intensity u in Erlang is given by

$$u = \frac{\lambda}{\mu}$$

The probability p_i for a number of i jobs in the system is given by

$$p_0 = \left[\sum_{n=0}^{c} \frac{u^n}{n!} + \frac{u^c}{c!} \sum_{n=1}^{k-c} \left(\frac{u}{c} \right)^n \right]^{-1}$$

$$p_n = \begin{cases} \frac{u^n}{n!} p_0 & n = 1,2,\ldots,c \\ \frac{u^c}{c!} \left(\frac{u}{c} \right)^{n-c} p_0 & n = c+1,\ldots,k \end{cases}$$

The server utilization ρ is given by

$$\rho = (1 - p_k) \frac{u}{c}$$

The average traffic rate λ_a in the system is

$$\lambda_a = (1 - p_k)$$

The expected number of requests in the queue $E[N_q]$ is

$$E[N_q] = \frac{u^c p_0 \frac{u}{c}}{c!(1 - \frac{u}{c})^2} \left[1 - (\tfrac{u}{c})^{k-c+1} - (k - c + 1)(\tfrac{u}{c})^{k-c} (1 - \tfrac{u}{c}) \right]$$

The expected number of requests in service $E[N_S]$ is

$$E[N_S] = \sum_{n=0}^{c-1} np_n + c\left(1 - \sum_{n=0}^{c-1} p_n\right)$$

The expected number of requests in the system E[N] is

$$E[N] = E[N_q] + E[N_S]$$

The expected waiting time W is

$$W = \frac{E[N]}{\lambda_a}$$

5.2.2 Dimensioning Example

The following example illustrates the dimensioning of a location presence server. The resulting blocking probability at the server shall be smaller than $1E^{-3}$.

First one has to choose the ratio r of notifications per subscription that is application dependent.

In order to find a reasonable value for the ration of notifications per subscription, the assumption in the following example is made, that "Push/Pull" style of LBS applications will outnumber "Tracking" style applications. It is furthermore assumed that in the duration of one subscription typically only one notification is sent. This represents a model where most LBS applications just request the current position of the user. This results in a notification to subscription rate of

$$r = \frac{1}{1}$$

A 1:1 notification to subscription rate leads to a total SIP request rate of

$$\lambda = 2\alpha$$

at the location presence server.

It is further assumed that the server can process 50 threads in parallel and thus setting

$$c = 50$$

The location presence server shall be able to queue 20 requests, if all processing threads are busy. Thus the number of request in the system k is

$$k = 20 + c = 70$$

The performance of the location presence server largely depends on the time it takes to process one SIP request. This processing time is largely influenced by the radio access network delay and the processing times in the mobile terminal. From our prototype implementation, typical presence processing times of about 2 seconds per request were measured. This results in a service rate μ of

$$\mu = \frac{1}{2}$$

The blocking probability P[B] is the probability that all service threads are busy and that the queue is full:

$$P[B] = p_k$$

Figure 5-7 shows the blocking probability of an M/M/c/k system with c=50 and k=70 over the traffic intensity.

For a requested blocking probability of $1E^{-3}$ the chart gives a traffic intensity u of about 42 Erlang.

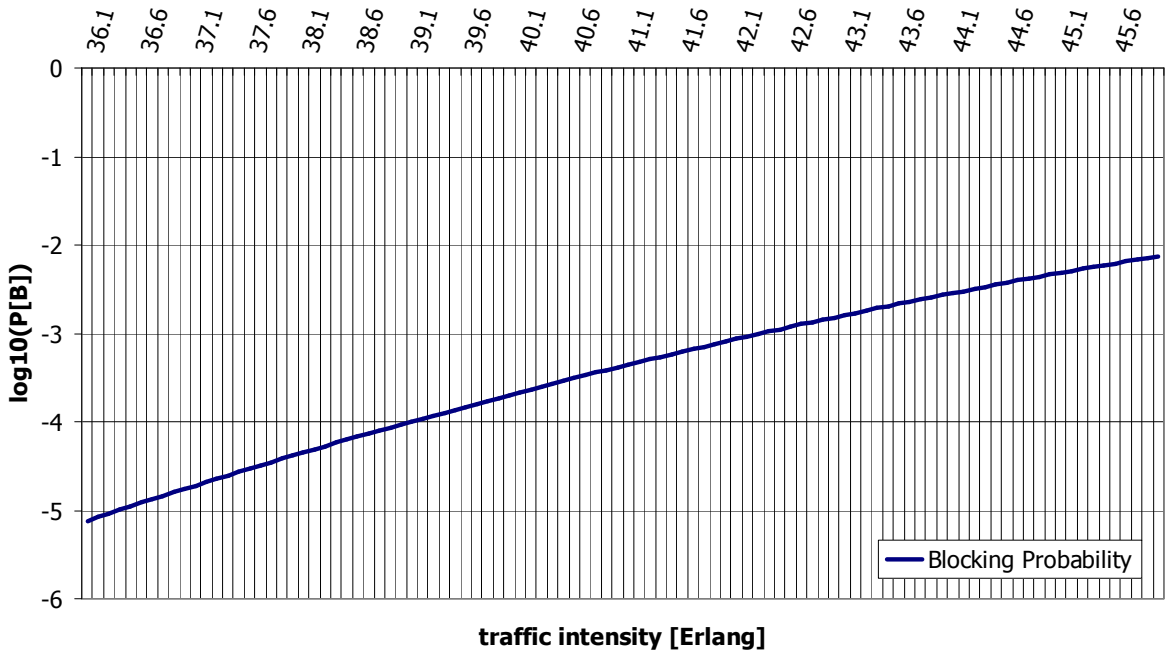


Figure 5-7: Blocking probability P[B] for an M/M/c/k queue with c=50 and k=70

This yields the following result. The measured (or estimated) mean service rate μ and the traffic intensity u from the blocking probability chart give the required maximum SIP request rate λ_{max}.

$u = 42$

$\mu = 0.5$

$\lambda_{\max} = u\mu = 21$

This means that the mean rate of SIP requests arriving at the server should not exceed 21 requests per second in order for the blocking probability to stay below $1E^{-3}$.

With a notification to subscription rate of r=1 this means that for a blocking probability less than 1E-3 one location presence server with an input queue of length 20 and at most 50 concurrent processes can accept SUBSCRIBE message at a rate of less than 10.5 location presence subscriptions per second.

When receiving 21 requests per second, the following performance parameters can be computed for the system.

$\rho = 0.84$

$E[N] = 42.72$

$W = 2.04$

It can be seen from the expected waiting time W and the expected number of requests in the system E[N], that the queue is nearly not used at all for a blocking probability less than $1E^{-3}$.

5.2.3 Availability Model for a Location Presence Server

In the section above the required maximum blocking probability was used to compute the maximum number of requests that the modelled server is able to process. The assumed blocking probability however holds only, if the server is always online.

A real server however is never available all the time, due to

- Scheduled downtimes for updates on hardware or software (e.g., releases, bug fixes)
- Unscheduled downtimes or failures or hardware and software

The concept of availability of a system is based on the notion that a system alternates between the states 'up' and 'down' [Menasce 2002]. The state 'up' means that the system is operational while 'down' means that the system is not functional. The average time is takes a system to fail is called Mean Time To Failure (MTTF). Once the system has failed, it takes a certain time, until it is operational again. The average time it takes for the system to recover from failure is called Mean Time To Recover (MTTR). The average time between failures is called Mean Time Between Failures (MTBF) and can be written as

$$MTBF = MTTF + MTTR$$

The relation ship between MTTF, MTTR and MTBF can be see in Figure 5-8.

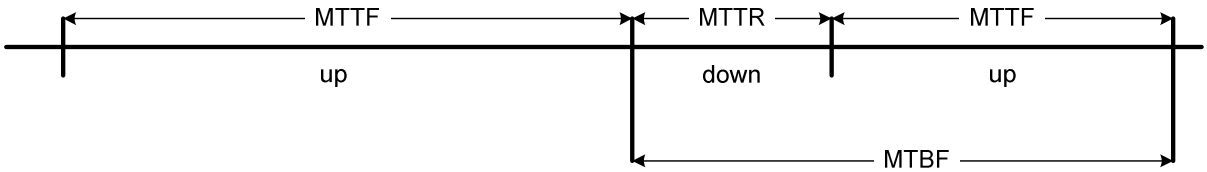


Figure 5-8: Relationship between Mean Time To Failure (MTTF), Mean Time To recovery (MTTR) and Mean Time Between Failures (MTBF)

The availability p[A] of a system is defined as the probability that the system is in state 'up' which is equivalent to the fraction of time that the system is operational.

If a server e.g., needs in average 30 min maintenance per week, it's availability results in

$$p[A] = 1 - (30 * 52) / (52 * 7 * 24 * 60) = 0.997$$

Taking a blocking probability of $1E^{-3}$ from the example in the previous section, the probability that the server is available and is not blocked is therefore

$$p[A \wedge \neg B] = p[A]p[\neg B \mid A] = p[A](1 - p[B]) = 0.997 * 0.999 = 0.996$$

From the perspective of the customer there is no difference, if a system is not responding because it is not operational or because a queue is blocked. In both cases the service cannot be provided. The probability however that the system is operational and that the queue is not blocked is smaller than each of the multiplicands in the equation above.

A general availability model shall be discussed based on the transition diagram shown in Figure 5-9.

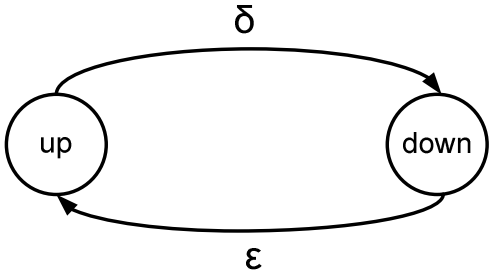


Figure 5-9: State transition diagram for the computation of system availability

The system fails or goes from state 'up' to state 'down' with a rate of δ. With a rate of ε the system gets repaired and goes from state 'down' back in state 'up'. These rates can be expressed in terms of MTTF and MTTR:

$$\delta = \frac{1}{MTTF}$$

$$\varepsilon = \frac{1}{MTTR}$$

If the system is in a statistical equilibrium, the flows in and out of a state must be equal:

$$\delta p_{up} = \varepsilon p_{down}$$

The probability that the system is up p_{up} and the probability that the system is down p_{down} add up to 1:

$$p_{up} + p_{down} = 1$$

The availability p[A] of the system can be calculated as

$$p[A] = p_{up} = \frac{\varepsilon}{\delta + \varepsilon} = \frac{MTTF}{MTTR + MTTF}$$

$$p_{down} = \frac{\delta}{\delta + \varepsilon} = \frac{MTTR}{MTTR + MTTF}$$

In most systems of interest, it takes significantly longer for the system to fail than to be repaired:

$$p_{down} \approx \frac{MTTR}{MTTF} \quad for \quad MTTF >> MTTR$$

There are two ways to improve the availability of a system: reduce the frequency of failures or reduce the time to recover from them. Changes in MTTR however have more influence on the availability than changes in MTTF:

$$p[A] = a = \frac{MTTF}{MTTR + MTTF} = \frac{F}{F + R}$$

$$\frac{da}{dR} = -\frac{F}{(F+R)^2} \quad F = const.$$

$$\frac{da}{dF} = \frac{R}{(F+R)^2} \quad R = const.$$

$$for\ F >> R$$

$$\left|\frac{da}{dR}\right| \approx \left|-\frac{1}{F}\right| > \left|\frac{da}{dF}\right| \approx \left|\frac{R}{F^2}\right|$$

It is a significant result from these equations that for improving system availability, it is more efficient to try to improve the system recovery times rather than to increase the time to the next failure.

Revisiting the example above, the Mean Time To Failure can be calculated for a given availability and a given mean recover time:

$$p[A] = a = 0.997$$

$$MTTR = 20\,\text{min}$$

$$MTTF = MTTR\frac{a}{1-a} = 6647\,\text{min} \approx 4.6\ days$$

5.2.4 Cluster Availability of a Location Presence Servers

The availability of a system can be increased by operating several servers in parallel. Figure 5-10 shows how a number of k servers are configured in parallel behind a load balancer.

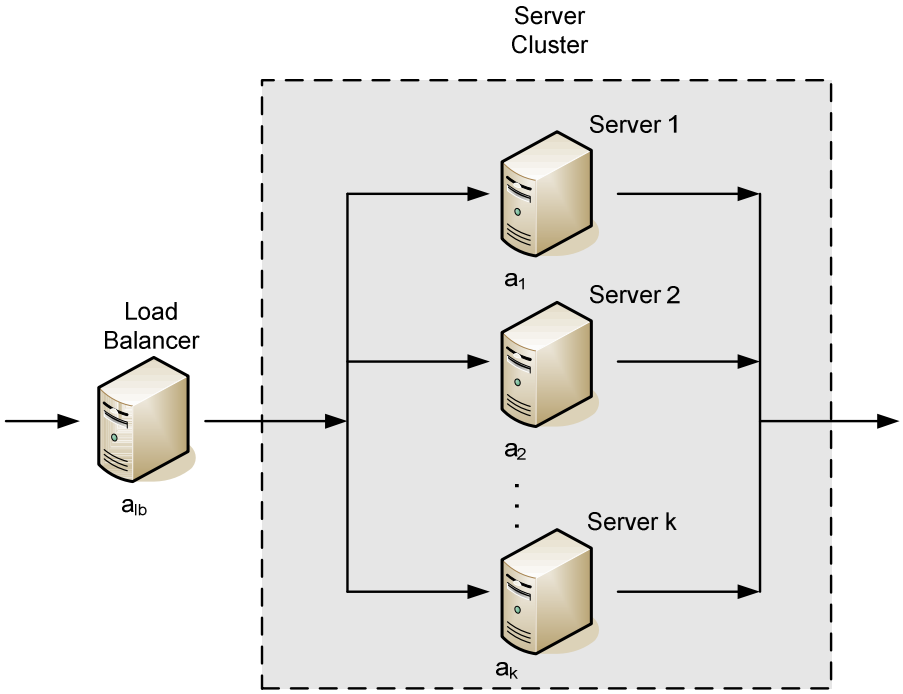


Figure 5-10: Configuration of servers in a cluster

A Load balancer distributes incoming requests to the servers in the cluster. The policy that chooses a server for the current request at the load balancer depends on the protocol and the application. In case of a Location Presence server the consistency of SIP transactions and SIP dialogues have to be preserved. This means that a SIP message that arrives at on of the physical server machines has to be processed in the context of the associated SIP transaction and SIP dialogue.

A SIP transaction is a SIP request sent by a User Agent Client (UAC) to a User Agent Server (UAS), along with all responses to that request sent from the UAS back to the UAC. A SIP transaction is identified by the Call-ID header, the CSeq header and the branch parameter of the Via header.

A SIP dialogue is a peer-to-peer relationship between two user agents. It represents a context that facilitates the sequencing of messages between the user agents and proper routing of requests between both of them. A SIP dialogue is identified by the Call-ID header, the CSeq header, the tag parameter of the To header, the tag parameter of the From header, the request URI and the URI set in the Route header.

A SIP aware Load Balancer typically distributes requests according to the SIP Call-ID header to the server machines. If a load balancer is used that is not SIP aware, the individual server has to implement some kind of mechanism that allows to reconstruct SIP transaction and SIP dialogue context. This can be achieved by persisting SIP context data to a database system or by distributing SIP context data among all physical machines.

The availability a_{sys} of a cluster of servers is calculated from the availability of the load balancer a_{lb} and the availability of the individual server a_i. The availability of n servers in parallel is

the probability that one or more servers are available. The n server availability is calculated from the probability that not all n servers fail simultaneously:

$$a_{sys} = a_{lb}\left(1 - \prod_{i=1}^{k}(1 - a_i)\right)$$

If all servers have the same availability a_s:

$$a_{sys} = a_{lb}\left(1 - (1 - a_s)^k\right)$$

For a large number of servers the system availability converges against the load balancer availability:

$$\lim_{k->\infty} a_{sys} = a_{lb}$$

The number of identical servers that are necessary to reach a given availability can be calculated as:

$$k = \frac{\ln\left(1 - \frac{a_{sys}}{a_{lb}}\right)}{\ln(1 - a_s)}$$

The last equation shows that there is a trade-off between the availability of a single server and the necessary number of servers. Server investment costs and costs of ownership have to be considered in order to find the optimal number of servers.

a_{sys}	a_{lb}	a_s	k
0,999	0,9999	0,85	3,7
0,999	0,9999	0,997	1,21

Table 5-1: required number of servers to meet a given system availability

Table 5-1 shows that 4 servers with low availability are needed to reach a given system availability of 0,999, whereas the system can also be built with 2 servers that have each a much higher availability.

5.2.5 Cluster Throughput of a Location Presence Server

The required number of servers in a cluster is not only influenced by the aggregate system availability, but also by the required system throughput.

Figure 5-11 shows the model of a cluster of k location presence servers. The total traffic with rate Λ is load balanced to k servers. A request gets forwarded by the load balancer to a server with a probability of 1/k. Each location presence server is modelled as an M/M/c/k queuing system.

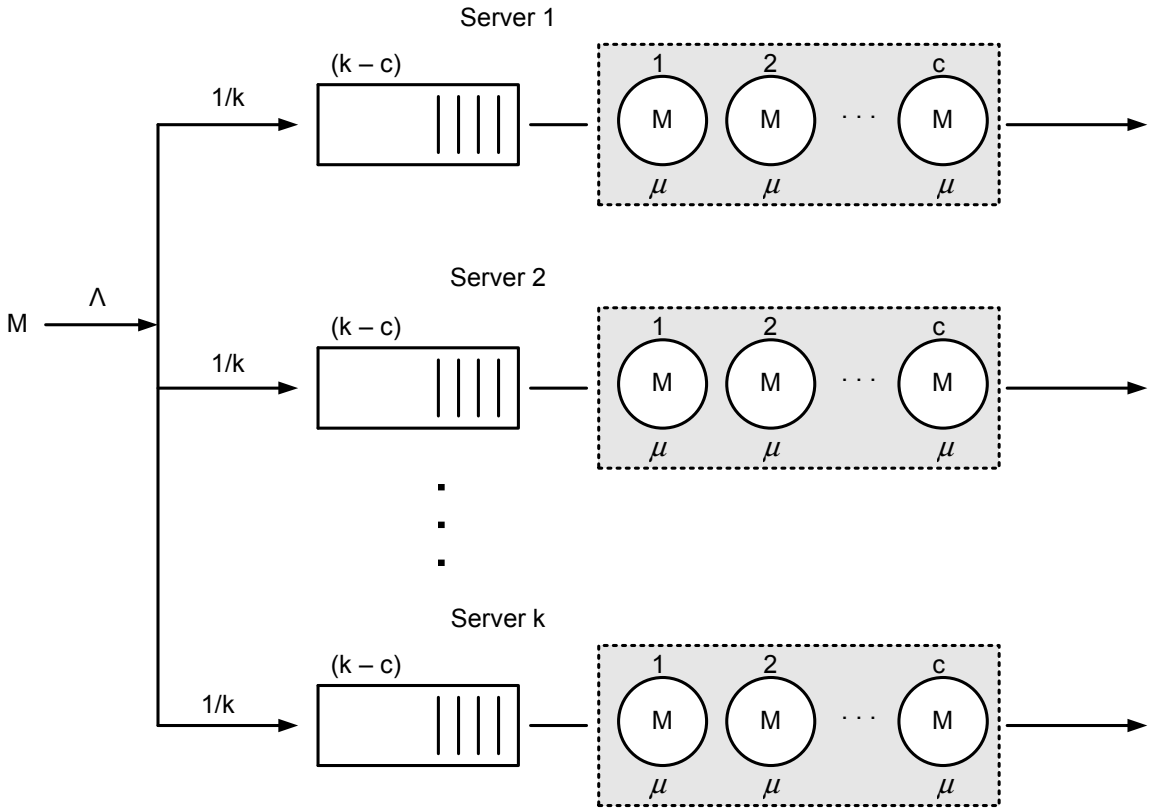


Figure 5-11: Cluster model

The average throughput λ_a at one server depends on the blocking probability and is a function of the traffic intensity u (see the analytical model above):

$$\lambda_a = \lambda_s(1 - P[B]) = \lambda_s(1 - f(\tfrac{u}{c}))$$

The total traffic rate Λ however is load balanced on k servers. This leads to a traffic rate λ_s at the individual server:

$$\lambda_s = \frac{\Lambda}{k}$$

The traffic intensity u depends on the individual traffic rate at the server and the service rate μ and thus on the number k of servers:

$$u = \frac{\lambda_s}{\mu} = \frac{\Lambda}{k\mu}$$

This means that the average throughput on one individual server depends on the number k of available servers in the cluster.

active servers k	traffic rate at individual server λ_s	service rate μ	traffic intensity u	blocking probability P[B]	average server throughput λ_a	cluster throughput
4	20,00	0,5	20,00	0,0002	19,99	79,96
3	26,67	0,5	53,33	0,074	24,69	74,07
2	40,00	0,5	80,00	0,38	24,99	49,98
1	80,00	0,5	160,00	0,69	25,00	25,00

Table 5-2: cluster throughput for different availability scenarios

Table 5-1 shows an example of the average server throughput and the resulting cluster throughput for all availability scenarios in a cluster of 4 servers. A total mean arrival rate Λ of 80 SIP requests / sec is assumed. The server model is taken from the examples above, where the maximum number of processing threads c at one server is 50 and the queue length in 20.
The blocking probability $P[B_C]$ of a cluster of k servers behind a load balancer is equal to the blocking probability of a single server P[B]:

$$P[B_C] = \sum_{i=1}^{k} \frac{1}{k} P[B] = \mathrm{P[B]}$$

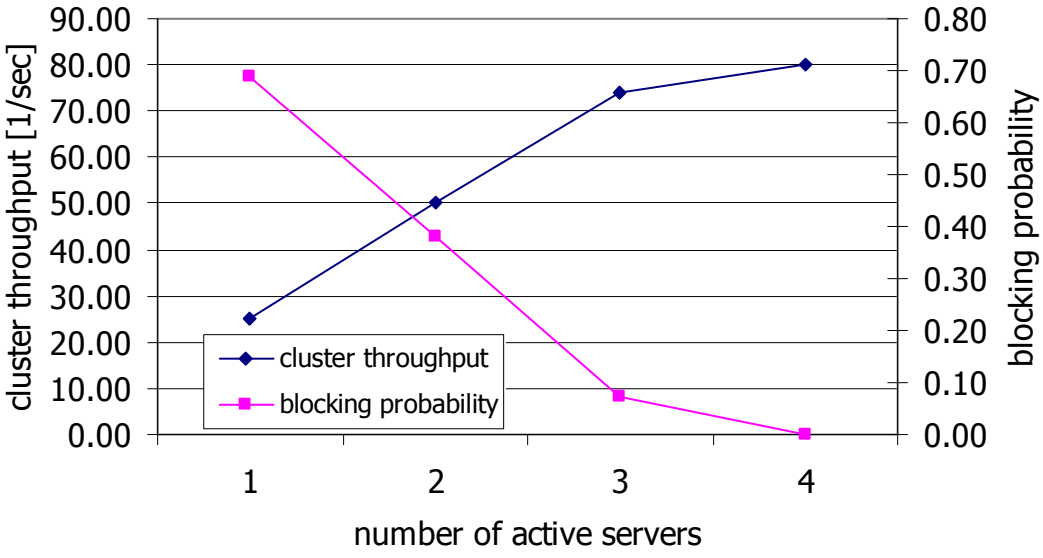


Figure 5-12: cluster throughput for a given number of active servers

It can be seen in Figure 5-12 that the blocking probability increases significantly and the cluster throughput decreases, if only 2 or less servers are available.

5.3 Presence Server with a Non-Preemptive Priority Queue

This section develops a system model for a Location Presence server with a non-preemptive priority queue. The idea of this system model is to analyze to influence of queue prioritization for different location presence applications on performance measures like expected number of jobs in the queues, waiting probabilities or blocking probabilities for jobs with different priorities.
First the model is discussed by means of a system with two different priorities (high/low). Then the equations are generalized to a system with n different priorities.

5.3.1 System with Two Queues

Figure 5-13 shows the system model of a non-preemptive priority queueing system with two queues. The jobs are generated from different applications for priorities 1 and 2. Priority 1 jobs arrive with a mean rate λ_1 and are characterized by a negative exponential interarrival time distribution. Jobs with priority 2 are characterized by a negative exponential interarrival time distribution with a mean rate of λ_2. The queue for jobs with priority 1 has L_1 places, the queue for priority 2 L_2 places. There is one service unit with negative exponential service time distribution with a mean service rate of μ. The service unit takes jobs from priority queue 1 with probability q and from priority queue 2 with probability (1-q).

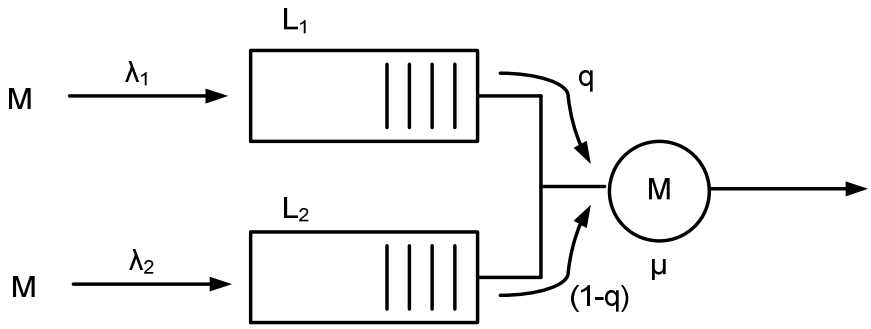


Figure 5-13: Non-preemptive priority queue model with two queues

In order to develop a system model based on a homogeneous Markov chain, the system is modelled by states where the time spent in state has a negative exponential distribution. From this a state transition probability matrix is derived. Figure 5-14 shows a general definition of system states.

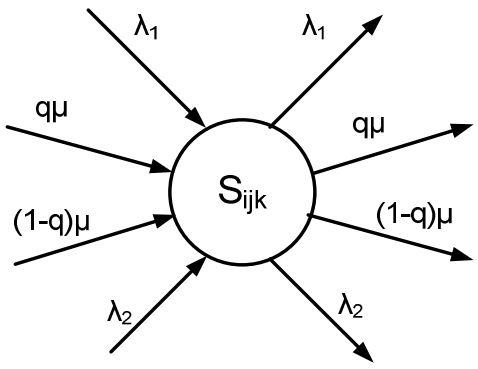


Figure 5-14: General definition of system states

The system states S_{ijk} are indexed by three indices i, j, k, denoting

- i: the number of jobs in the queue for priority 1
- j: the number of jobs in the queue for priority 2
- k: server state
  - k = 0: server is empty
  - k = 1: priority 1 job in service
  - k = 2: priority 2 job in service
  - K = 3: number of server states

In general, events that trigger a state transition can happen with mean rates λ_1, λ_2, μq and $\mu(1-q)$. This is shown in Figure 5-14 as arrows entering state S_{ijk} or departing from state S_{ijk}.
In the following, the algorithm for the state transitions is developed, that describes the model in Figure 5-13:

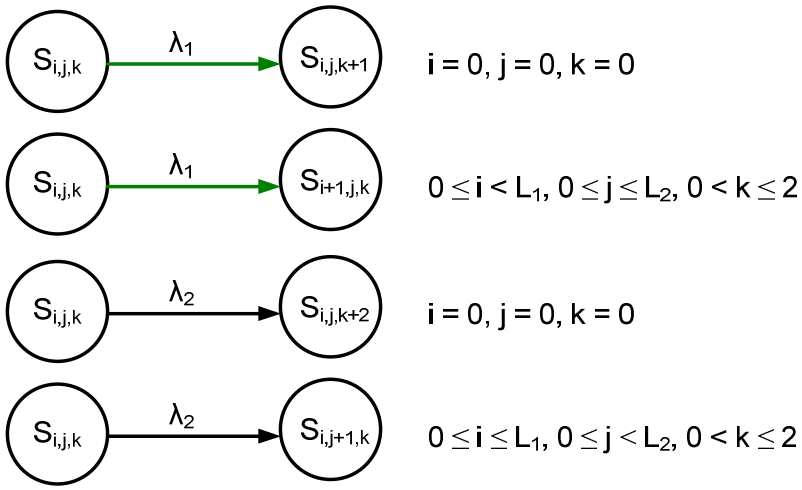


Figure 5-15: state transitions of the arrival process

The state transitions for the arrival process are shown in Figure 5-15. Arrivals with priority 1 happen with a mean rate of λ_1 and arrivals with priority 2 happen with a mean rate of λ_2:

- An arrival with priority 1 happens when the system is empty (i = 0, j = 0, k = 0). This event with priority 1 is immediately processed by the service unit and thus state (i = 0, j = 0, k = 1) is entered.
- An arrival with priority 1 happens when the service unit is busy with either a job of priority 1 or a job of priority 2 (k = 1, 2). Then the job is put in the priority 1 queue (i is incremented by 1) as long as the queue has free places ($0 \le i < L_1$). Otherwise the system is blocked for priority 1 events.
- An arrival with priority 2 happens when the system is empty (i = 0, j = 0, k = 0). This event with priority 2 is immediately processed by the service unit and thus state (i = 0, j = 0, k = 2) is entered.
- An arrival with priority 2 happens when the service unit is busy with either a job of priority 1 or a job of priority 2 (k = 1,2). Then the job is put in the priority 2 queue (j is incremented by 1) as long as the queue has free places ($0 \le j < L_2$). Otherwise the system is blocked for priority 2 events.

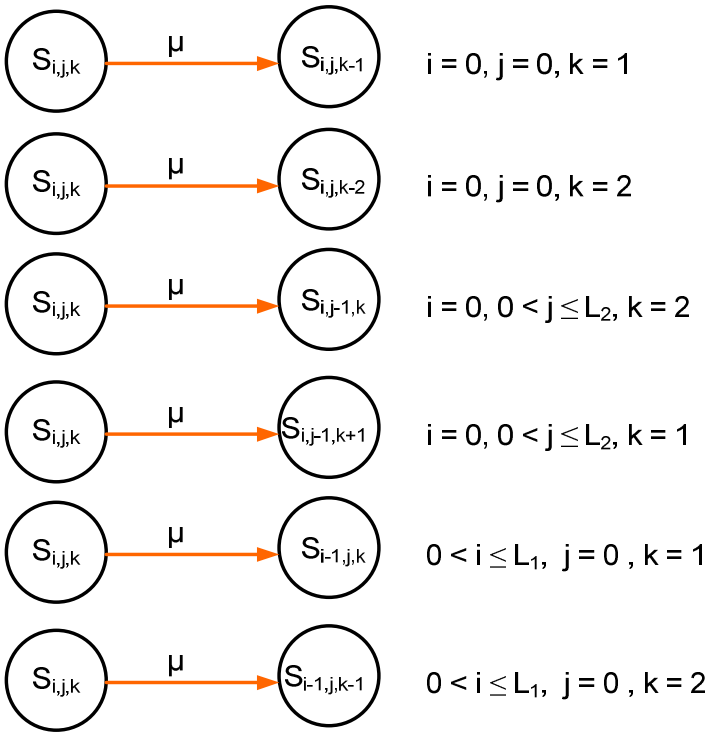


Figure 5-16: deterministic state transitions of the service process

The deterministic state transitions of the service process are shown in Figure 5-16. Service process events happen with a mean rate of μ:

- The service unit processes a job of priority 1 (k = 1) and the queues are empty (i = 0, j = 0). Then the empty state (i = 0, j = 0, k = 0) is entered.
- The service unit processes a job of priority 2 (k = 2) and the queues are empty (i = 0, j = 0). Then the empty state (i = 0, j = 0, k = 0) is entered.
- The queue for priority 1 is empty (i = 0), the queue for priority 2 is not empty ($0 < j \leq L_2$) and the service unit processes a job of priority 2 (k = 2). Then a job is taken out of queue for priority 2 (j is decremented by 1 and k is set to 2).
- The queue for priority 1 is empty (i = 0), the queue for priority 2 is not empty ($0 < j \leq L_2$) and the service unit processes a job of priority 1 (k = 1). Then a job is taken out of queue for priority 2 (j is decremented by 1 and k is set to 2).
- The queue for priority 2 is empty (j = 0), the queue for priority 1 is not empty ($0 < i \leq L_1$) and the service unit processes a job of priority 1 (k = 1). Then a job is taken out of queue for priority 1 (i is decremented by 1 and k is set to 1).
- The queue for priority 2 is empty (j = 0), the queue for priority 1 is not empty ($0 < i \leq L_1$) and the service unit processes a job of priority 2 (k = 2). Then a job is taken out of queue for priority 1 (i is decremented by 1 and k is set to 1).

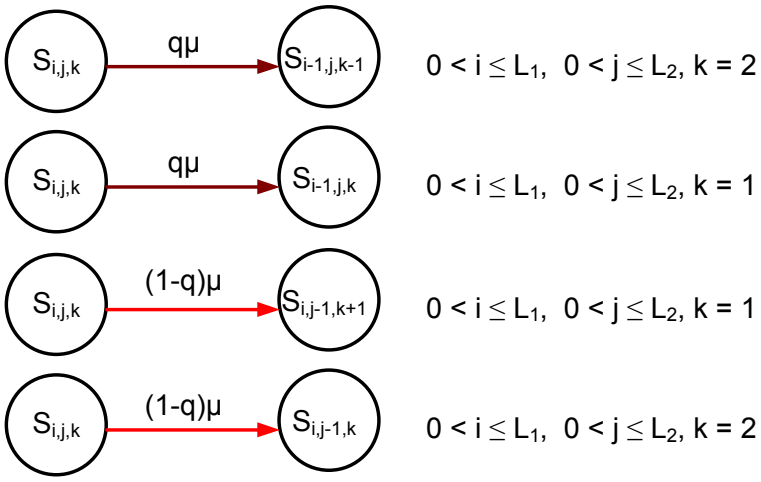


Figure 5-17: random policy state transitions of the service process

The random policy state transitions of the service process are shown in Figure 5-17. Service events for priority 1 queue happen with a mean rate of qμ, while service events for priority 2 queue happen with a mean rate of (1-q)μ. The random queue policy is controlled by the probability q that a job is taken out of the priority 1 queue:

- Both queues are not empty ($0 < i \leq L_1$, $0 < j \leq L_2$) and the server processes a job with priority 2 (k = 2). With a probability q a job is taken from priority 1 queue(i is decremented by 1 and k is set to 1).
- Both queues are not empty ($0 < i \leq L_1$, $0 < j \leq L_2$) and the server processes a job with priority 1 (k = 1). With a probability q a job is taken from priority 1 queue(i is decremented by 1 and k is set to 1).
- Both queues are not empty ($0 < i \leq L_1$, $0 < j \leq L_2$) and the server processes a job with priority 1 (k = 1). With a probability (1-q) a job is taken from priority 2 queue(j is decremented by 1 and k is set to 2).
- Both queues are not empty ($0 < i \leq L_1$, $0 < j \leq L_2$) and the server processes a job with priority 2 (k = 2). With a probability (1-q) a job is taken from priority 2 queue(j is decremented by 1 and k is set to 2).

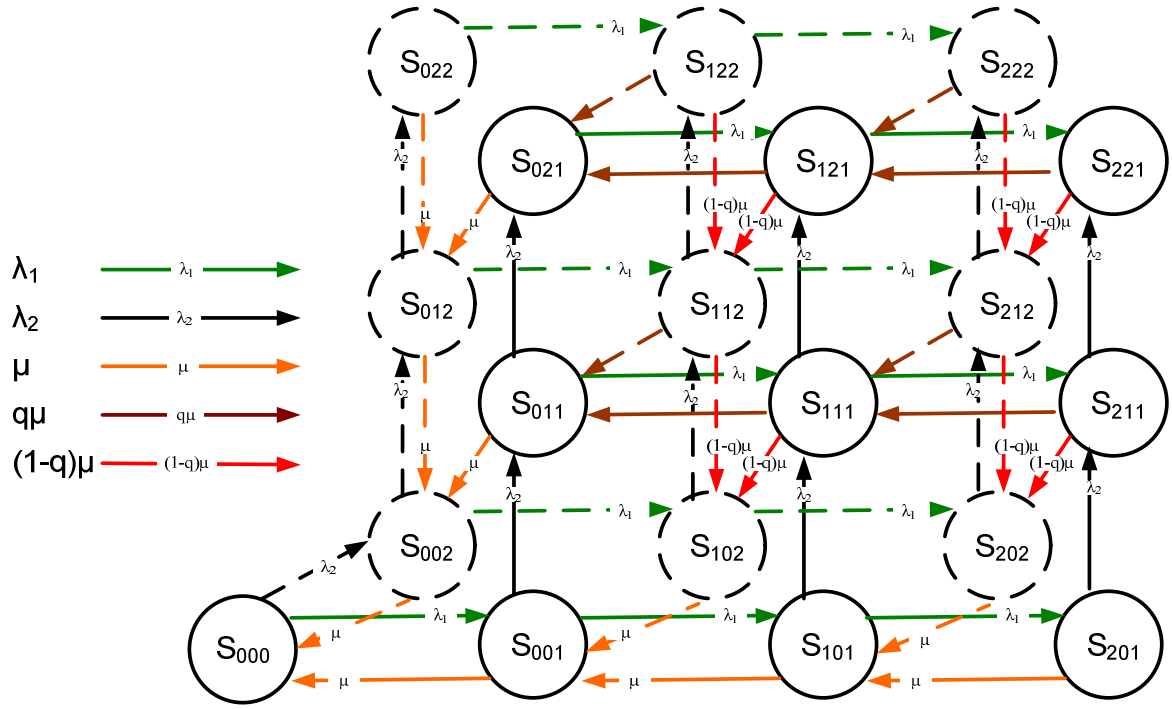


Figure 5-18: State transition diagram of a non-preemptive priority queueing system with two queues

Figure 5-18 shows the state transition diagram of a non-preemptive priority queueing system with two queues and queue lengths of $L_1 = 2$ and $L_2 = 2$. It can be easily seen that for larger queue lengths, the diagram has to be extended in the indices i and j. This means that the diagram would grow to the top and to the right.

The state probability equations in statistical equilibrium describe all possible state transitions in the system, where P_{ijk} denotes the probability of the system to be in state S_{ijk}:

$$P_{ijk} := P(S_{ijk})$$

$$\begin{aligned}
(\mu + \mu)\ P_{000} &= (\lambda_1)\ P_{001} + (\lambda_2)\ P_{002} \\
(\lambda_1 + \mu + \mu)\ P_{001} &= (\mu)\ P_{000} + (\lambda_2)\ P_{011} + (\lambda_1)\ P_{101} \\
(\lambda_2 + \mu + \mu)\ P_{002} &= (\mu)\ P_{000} + (\lambda_2)\ P_{012} + (\lambda_1)\ P_{102} \\
(\lambda_2 + q\mu + q\mu)\ P_{011} &= (\mu)\ P_{002} + (\lambda_2)\ P_{021} + (\lambda_1)\ P_{111} \\
(\lambda_2 + \mu + \mu)\ P_{012} &= (\mu)\ P_{002} + (\lambda_2)\ P_{022} + (\lambda_1)\ P_{112} \\
(\lambda_2 + q\mu + q\mu)\ P_{021} &= (\mu)\ P_{012} + (\lambda_1)\ P_{121} \\
(\lambda_2)\ P_{022} &= (\mu)\ P_{012} + (\lambda_1)\ P_{122} \\
(\lambda_1 + \mu + \mu)\ P_{101} &= (\mu)\ P_{001} + (\lambda_2)\ P_{111} + (\lambda_1)\ P_{201} \\
(\lambda_1 + (1\text{-}q)\mu + (1\text{-}q)\mu)\ P_{102} &= (\mu)\ P_{001} + (\lambda_2)\ P_{112} + (\lambda_1)\ P_{202} \\
(\lambda_1 + \lambda_2 + q\mu + q\mu)\ P_{111} &= (q\mu)\ P_{011} + ((1\text{-}q)\mu)\ P_{102} + (\lambda_2)\ P_{121} + (\lambda_1)\ P_{211} \\
(\lambda_1 + \lambda_2 + (1\text{-}q)\mu + (1\text{-}q)\mu)\ P_{112} &= (q\mu)\ P_{011} + ((1\text{-}q)\mu)\ P_{102} + (\lambda_2)\ P_{122} + (\lambda_1)\ P_{212} \\
(\lambda_1 + \lambda_2 + q\mu + q\mu)\ P_{121} &= (q\mu)\ P_{021} + ((1\text{-}q)\mu)\ P_{112} + (\lambda_1)\ P_{221} \\
(\lambda_1 + \lambda_2)\ P_{122} &= (q\mu)\ P_{021} + ((1\text{-}q)\mu)\ P_{112} + (\lambda_1)\ P_{222} \\
(\lambda_1)\ P_{201} &= (\mu)\ P_{101} + (\lambda_2)\ P_{211} \\
(\lambda_1 + (1\text{-}q)\mu + (1\text{-}q)\mu)\ P_{202} &= (\mu)\ P_{101} + (\lambda_2)\ P_{212} \\
(\lambda_1 + \lambda_2)\ P_{211} &= (q\mu)\ P_{111} + ((1\text{-}q)\mu)\ P_{202} + (\lambda_2)\ P_{221} \\
(\lambda_1 + \lambda_2 + (1\text{-}q)\mu + (1\text{-}q)\mu)\ P_{212} &= (q\mu)\ P_{111} + ((1\text{-}q)\mu)\ P_{202} + (\lambda_2)\ P_{222} \\
(\lambda_1 + \lambda_2)\ P_{221} &= (q\mu)\ P_{121} + ((1\text{-}q)\mu)\ P_{212} \\
(\lambda_1 + \lambda_2)\ P_{222} &= (q\mu)\ P_{121} + ((1\text{-}q)\mu)\ P_{212}
\end{aligned}$$

This system of linear equations can be written as a matrix equation. In order to solve the equations with a standard linear algebra package, the state indices i, j, k are transformed to a single index x:

$$\mathbf{p} = \{P_{ijk}\} \quad 0 \le i \le L_1, 0 \le j \le L_2, 0 \le k < K, \quad i, j, k \in \aleph_0$$

$$\mathbf{p} = \{p_x\} \quad x = i(L_1 + 1)(L_2 + 1) + j(L_2 + 1) + k$$

$$\mathbf{A}\mathbf{p} = 0$$

Matrix **A** has the following properties with n denoting the number of states:

$$\dim(\mathbf{A}) = n \times n$$

$$n = (L_1 + 1)(L_2 + 1)K$$

$$rank(\mathbf{A}) = n - 1$$

The rank of Matrix **A** is by 1 smaller than the dimension of the matrix. This means that the system of equations is linearly dependent and that that matrix **A** is singular.

The definition of a probability measure furthermore requires that:

$$\sum_{i,j,k} P_{ijk} = 1$$

In order to solve the system of linear equations this probability normalization condition has to be integrated.

There are two ways to integrate the probability normalization condition into the system of linear equations. Figure 5-19 shows how matrix **A** can be extended by one additional row. This however leads to non-square matrix **A**.

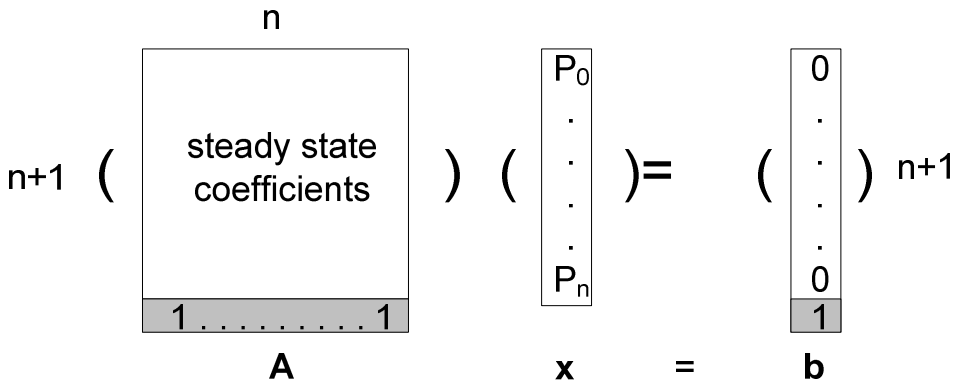


Figure 5-19: steady state probability equations, extended by one row for the probability normalization condition

Figure 5-20 shows that one row of matrix **A** can be replaced by the probability normalization condition. This follows from the fact, that the rank of Matrix A is n-1, meaning that out of n rows, one is redundant and can be replaced without changing the solution of the system.

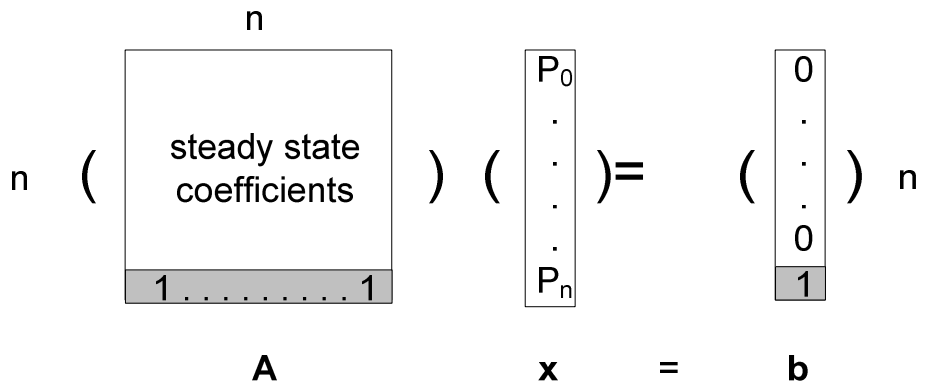


Figure 5-20: steady state probability equations, one row replaced y the probability normalization condition

This matrix equation, representing a system of linear equations can be solved for the individual state probabilities in statistical equilibrium.

The following performance measures can then be calculated from the state probabilities.

Expected value for the number of jobs in the system X:

$$E[X] = \left(\sum_{i=0}^{L_1} \sum_{j=0}^{L_2} \sum_{k=1}^{K} (i + j + 1) P_{ijk} \right) + \left(\sum_{i=0}^{L_1} \sum_{j=0}^{L_2} (i + j) P_{ij0} \right)$$

The mean system throughput T is the mean service rate multiplied with the probability that the server is busy:

$$T = \mu \left(\sum_{i=0}^{L_1} \sum_{j=0}^{L_2} \sum_{k=1}^{K} P_{ijk} \right) = \mu (1 - P_{000})$$

The blocking probability for jobs with priority 1 $P(B_1)$ is the sum of all state probabilities, where the queue for priority 1 is full:

$$P(B_1) = \sum_{j=0}^{L_2} \sum_{k=0}^{K} P_{L_1 jk}$$

The waiting probability for jobs with priority 1 $P(W_1)$ is the sum of all state probabilities, where the server is busy and queue 1 is not blocked:

$$P(W_1) = \sum_{i=0}^{L_1 - 1} \sum_{j=0}^{L_2} \sum_{k=1}^{K} P_{ijk}$$

The following example shows the principles of calculating the steady state probabilities:

- Queue length priority 1: $L_1 = 2$
- Queue length priority 2: $L_2 = 2$
- Arrival rate priority 1, $\lambda 1 = 2.0$
- Arrival rate priority 2, $\lambda 2 = 4.0$
- Service rate, $\mu = 8.0$
- Probability for service of priority 1 job, if both queues are full, $q = 0.8$
- Probability for service of priority 2 job, if both queues are full, $1\text{-}q = 0.2$

This example was implemented in Java, using CERN's COLT ‚Open Source Libraries for High Performance Scientific and Technical Computing in Java'. On the COLT homepage (http://dsd.lbl.gov/~hoschek/colt/) it is noted that "Scientific and technical computing, as, for example, carried out at CERN, is characterized by demanding problem sizes and a need for high performance at reasonably small memory footprint. There is a perception by many that the Java language is unsuited for such work. However, recent trends in its evolution suggest that it may soon be a major player in performance sensitive scientific and technical computing. For example, IBM Watson's Ninja project showed that Java can indeed perform BLAS matrix computations up to 90% as fast as optimized Fortran[...].The reasons include ease of use, cross-platform nature, built-in support for multi-threading, network friendly APIs and a healthy pool of available developers".

Note well, that not all states S_{ijk} are used in the system. All states where at least one of the queues in not empty, but the server is empty are not valid in the model and have therefore a probability of 0:

$$\forall S_{ijk} : (i \neq 0 \vee j \neq 0) \wedge k = 0 \Rightarrow P_{ijk} = 0$$

The dimension of matrix **A** is thus reduced by the number of not reachable states, as these states only lead to rows and columns filled by zeros.

The program output for the solution of the steady state probabilities can be found in *Appendix B: Program* .

This gives the following steady state probabilities:

P_{000} = 0.33685533360234615
P_{001} = 0.07481054524112882
P_{002} = 0.1778309549606308
P_{101} = 0.016138174102549124
P_{102} = 0.030566446668839776
P_{201} = 0.002689695683758186
P_{202} = 0.006849472685420573
P_{011} = 0.04202253962389262
P_{012} = 0.1007539647560382
P_{111} = 0.016631492229103643
P_{112} = 0.02853622242245595
P_{211} = 0.0036684805994366696
P_{212} = 0.009494506205167905
P_{021} = 0.04710237494033617
P_{022} = 0.04030158590241528
P_{121} = 0.027858567429876958
P_{122} = 0.019474806149465437
P_{221} = 0.008798882157187575
P_{222} = 0.00961595463995034

With these state probabilities the following performance parameters can be calculated:

Expected value for the number of jobs in the system X:

$$E[X] = \left(\sum_{i=0}^{L_1} \sum_{j=0}^{L_2} \sum_{k=1}^{K} (i+j+1) P_{ijk} \right) + \left(\sum_{i=0}^{L_1} \sum_{j=0}^{L_2} (i+j) P_{ij0} \right) = 1.392$$

The mean system throughput T is the mean service rate multiplied with the probability that the server is busy:

$$T = \mu \left(\sum_{i=0}^{L_1} \sum_{j=0}^{L_2} \sum_{k=1}^{K} P_{ijk} \right) = \mu (1 - P_{000}) = 5{,}31$$

The blocking probability for jobs with priority x $P(B_x)$ is the sum of all state probabilities, where the queue for priority x is full:

$$P(B_1) = \sum_{j=0}^{L_2} \sum_{k=0}^{K} P_{L_1 jk} = 0{,}041$$

$$P(B_2) = \sum_{i=0}^{L_1} \sum_{k=0}^{K} P_{iL_2 k} = 0{,}15$$

The waiting probability for jobs with priority x $P(W_x)$ is the sum of all state probabilities, where the server is busy and queue x is not blocked:

$$P(W_1) = \sum_{i=0}^{L_1 - 1} \sum_{j=0}^{L_2} \sum_{k=1}^{K} P_{ijk} = 0{,}62$$

$$P(W_2) = \sum_{i=0}^{L_1} \sum_{j=0}^{L_2 - 1} \sum_{k=1}^{K} P_{ijk} = 0{,}51$$

5.3.2 General Non-Preemptive Priority Queueing System

The section develops a generalized system model for a non-preemptive priority queueing system. Compared to the system model presented in the previous section, this model has an arbitrary number of priorities and an arbitrary number of servers.

Figure 5-21 shows the system model of a general non-preemptive priority queueing system. The jobs are generated from different applications for priorities 1 to n. Jobs with priority i arrive with a mean rate λ_i and are characterized by a negative exponential interarrival time distribution. The queue for jobs with priority i has L_i places. There are m service units with negative exponential service time distribution with a mean service rate of μ_1. A service unit takes jobs from priority queue i with probability q_i on condition that the job is not taken from a queue with higher priority.

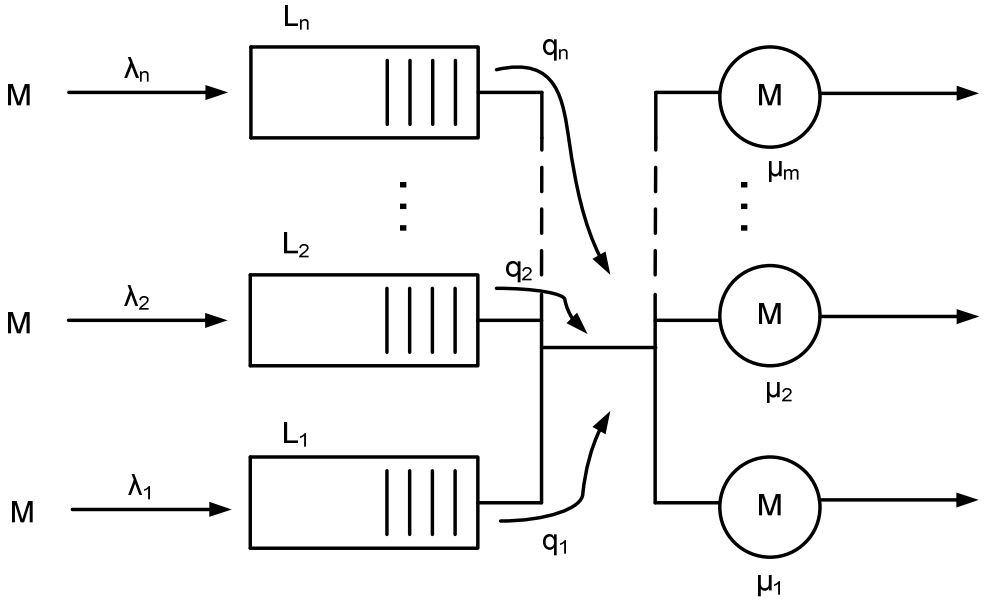


Figure 5-21: Model of a general non-preemptive priority queueing system

Figure 5-22 shows the system state definition for a general non-preemptive priority queueing system. The system state is composed of the state of all queues and the state of all servers.

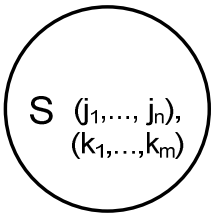


Figure 5-22: system state definition for a general non-preemptive priority queueing system

The system state represents the state of n queues and m servers. A queue with priority i can be empty or hold up to L_i jobs. The server can be empty or process a job from queue i.
The system state is defined as:

$S_{(j_1,\ldots,j_n),(k_1,\ldots,k_m)}$

n number of queues / priorities

$i: 1 \le i \le n$ priority index (lower index means higher priority)

L_i number of places in queue i

$j_i: 0 \le j_i \le L_i$ number of jobs in queue i

m number of servers

$l: 1 \le l \le m$ server index

$k_l: 0 \le k_l \le n$ priority of job in server l

$k_l = 0:$ server l is empty

$k_l = i:$ job with priority i in server l

$$N = \left[\prod_{i=1}^{n}(L_i + 1)\right] \cdot (n+1)^m \quad \text{number of states}$$

In the following, the state transitions for all arrival and service events are listed:

- Arrival of a job with priority x, when at least one server is empty:
  $\forall i : 1 \le i \le n, j_i = 0$
  $\exists l : 1 \le l \le m, k_l = 0$
  transition with rate λ_x :
  $k_l \rightarrow k_l = x$
- Arrival of a job with priority x, when all servers are busy
  $j_x < L_x$
  $\forall l : 1 \le l \le m, k_l \neq 0$
  transition with rate μ_y :
  $j_x \rightarrow j_x + 1$
- Service event of server y, when all queues are empty
  $\forall i : 1 \le i \le n, j_i = 0$
  $k_y \neq 0$
  transition with rate μ_y :
  $k_y \rightarrow k_y = 0$
- Service event of server y, when at least on queue is not empty
  $\exists i : 1 \le i \le n, j_i \neq 0$
  $k_y \neq 0$
  $p_i = P(\text{job taken from queue i})$
  $q_i = P(\text{job taken from queue i} \mid \text{job is not taken from a higher priority queue r, r < i})$

$$w_i = \begin{cases} 0 & j_i = 0 \\ q_i & i \neq \max(i : j_i \neq 0) \\ 1 & i = \max(i : j_i \neq 0) \end{cases}$$

$$p_i = \begin{cases} w_i \prod_{x=1}^{i-1} (1 - w_x) & 1 < i \le n \\ w_i & i = 1 \end{cases}$$

$$\sum_{i=1}^{n} p_i = 1$$

transition with rate $p_i \mu_y$:

$j_i \rightarrow j_i - 1$

$k_y \rightarrow k_y = i$

Note well that the probabilities p_i depend on the number of queues with different priorities that have jobs waiting. Only those queues that are occupied are relevant for the state transition. The probabilities p_i must sum up to 1, in order to fully specify the system. The probabilities q_i are static conditional probabilities for taking a job out of queue i, if a queue with higher priority has not be chosen and queues with lower priorities than i are occupied. The probabilities w_i reflect that the probability to choose an empty queue must be 0 and that the probability to choose the occupied queue with the lowest priority must be 1.

The probability that the system in a particular state is:

$$P_{(j_1 \ ,\ldots,j_n),(k_1,\ldots,k_m)} := P(S_{(j_1 \ ,\ldots,j_n),(k_1,\ldots,k_m)})$$

$$\sum P_{(j_1 \ ,\ldots,j_n),(k_1,\ldots,k_m)} = 1$$

Note well that there are not reachable states in the system At least one server must be busy, when at least one queue is not empty. Thus the state probability is 0 for all states where:

$$\left.\begin{array}{l}\exists i : 1 \le i \le n, j_i \ne 0 \\ \forall l : 1 \le l \le m, k_l = 0\end{array}\right\} P_{(j_1 \ ,\ldots,j_n),(k_1,\ldots,k_m)} = 0$$

In order to find a solution for the state probabilities in statistical equilibrium, the state transition matrix must be formed. For the implementation of the system of linear equations, a single numerical index z must be assigned to every state index:

$$S_z = S_{(j_1 \ ,\ldots,j_n),(k_1,\ldots,k_m)}$$

$$z = \left[j_1 + \sum_{i=2}^{n} j_i \left(\prod_{x=1}^{i-1} (L_x + 1) \right) \right] (n+1)^m + \left[\sum_{l=1}^{m} k_l (n+1)^{l-1} \right]$$

After solving the system of linear equation for the steady state probabilities, the following performance measures can be calculated:

- Blocking probability for a job with priority x:

$$P(B_x) = P(\text{queue x is full}) = \sum_{\substack{1 \le j_i \le n, i \ne x, 0 \le i \le L_i \\ j_x = L_x, i = x \\ 1 \le k_l \le m, 0 \le l \le n}} P_{(j_1 \ ,\ldots,j_n),(k_1,\ldots,k_m)}$$

- Waiting probability for a job with priority x:

$$P(W_x) = P(\text{queue x is not full and all servers busy}) =$$

$$\sum_{\substack{1 \le j_i \le n, i \ne x, 0 \le i \le L_i \\ j_x < L_x, i = x \\ 1 \le k_l \le m, k_l > 0, 0 \le l \le n}} P_{(j_1 \ ,\ldots,j_n),(k_1,\ldots,k_m)}$$

- Expected number of jobs in queue with priority x:

$$E[X_x] = \sum_{\substack{1 \le j_i \le n, 0 \le i \le L_i \\ 1 \le k_l \le m, 0 \le l \le n}} j_x P_{(j_1 \ ,\ldots,j_n),(k_1,\ldots,k_m)}$$

- Expected number of jobs in system:

$$E[X] = \sum_{\substack{1 \le j_i \le n, 0 \le i \le L_i \\ 1 \le k_l \le m, 0 \le l \le n}} \left(\sum_{x=1}^{n} j_x + \sum_{1 \le y \le m, k_y > 0} 1 \right) P_{(j_1 \ ,\ldots,j_n),(k_1,\ldots,k_m)}$$

- Mean service rate for priority x:

$$T_X = \sum_{\substack{1 \le j_i \le n, 0 \le i \le L_i \\ 1 \le k_l \le m, k_l = x, 0 \le l \le n}} \mu_l P_{(j_1 \ ,\ldots,j_n),(k_1,\ldots,k_m)}$$

- Mean service rate of system:

$$T = \sum_{\substack{1 \le j_i \le n, 0 \le i \le L_i \\ 1 \le k_l \le m, k_l > 0, 0 \le l \le n}} \mu_l P_{(j_1 \ ,\ldots,j_n),(k_1,\ldots,k_m)}$$

5.3.3 Dimensioning Example

The following example analyses a system with 2 priority queues (n = 2). The blocking probability for jobs with priority 1 shall be smaller than 10^{-3}. The blocking probability for jobs with priority 2 shall be smaller than 10^{-2}.
Jobs with priority i arrive at the system with a rate of λ_i.
The system shall be analysed for a different number of servers m. Each server shall have the same service rate μ.
The traffic intensity u is defined as:

$$u = \frac{\sum_{i=1}^{n} \lambda_i}{m \mu}$$

The traffic rates intensity shall be varied in the simulation from 0.1 to 1. The rates are chosen in the following way:

$$m\mu = 1$$

$$\lambda_1 + \lambda_2 = um\mu = u$$

$$\lambda_1 = \frac{u}{10}$$

$$\lambda_2 = u - \lambda_1 = \frac{9}{10}u$$

This means that 10% of the traffic has priority 1.
Note also, that by keeping mμ = 1, the total throughput of a multi threading system is simulated. The maximum server throughput is always kept constant at normalized value of 1, while this traffic throughput is divided among the simulated server processes. Thus the influence of having multiple server threads in parallel is analysed.
When all servers are busy, jobs with priority i are stored in a queue with L_i places. A job that arrives at a full queue is rejected and the system is called blocked for that priority. The queue lengths in this example are set to:

$$L_1 = 5$$

$$L_2 = 10$$

A server that has finished the current job takes a new job from one of the queues, if this queue is not empty. The highest priority i is chosen with a probability of q_i, if lower queues are not empty. In this example these probabilities are:

$$q_1 = 0.8$$

$$q_2 = 1.0$$

This means that if both queues are full a job is taken from queue 1 with probability 0.8.

Table 5-3 shows the performance evaluation results of the system with one server.

u	E[X]	E[Q1]	E[Q2]	T	Tprio[1]	Tprio[2]	P(B1)	P(B2)	P(W1)	P(W2)
0.1	0.1111	0.0010	0.0101	0.1000	0.0100	0.0900	1.3357E-11	4.6576E-12	0.1000	0.1000
0.2	0.2500	0.0044	0.0456	0.2000	0.0200	0.1800	1.0456E-09	1.0706E-08	0.2000	0.2000
0.3	0.4286	0.0102	0.1183	0.3000	0.0300	0.2700	1.3758E-08	9.4245E-07	0.3000	0.3000
0.4	0.6665	0.0189	0.2477	0.4000	0.0400	0.3600	8.5850E-08	2.1223E-05	0.4000	0.4000
0.5	0.9976	0.0305	0.4671	0.4999	0.0500	0.4499	3.5360E-07	2.2203E-04	0.4999	0.4997
0.6	1.4769	0.0455	0.8322	0.5992	0.0600	0.5392	1.1157E-06	1.3997E-03	0.5992	0.5978
0.7	2.1780	0.0636	1.4182	0.6962	0.0700	0.6262	2.9160E-06	6.0515E-03	0.6962	0.6901
0.8	3.1536	0.0845	2.2829	0.7862	0.0800	0.7062	6.5923E-06	1.9210E-02	0.7862	0.7670
0.9	4.3632	0.1070	3.3940	0.8622	0.0900	0.7722	1.3218E-05	4.6715E-02	0.8621	0.8154
1	5.6472	0.1296	4.5991	0.9185	0.1000	0.8185	2.3950E-05	9.0557E-02	0.9185	0.8279

Table 5-3: Performance measures of a priority queue system with 2 priorities and 1 server

The graph of the blocking probabilities Figure 5-23 shows that for a traffic intensity below 0.7, the blocking probability for jobs with priority 2 stays below the required limit of 10^{-2}. The blocking probability for jobs with priority 1 always stays below the required limit of 10^{-3}.

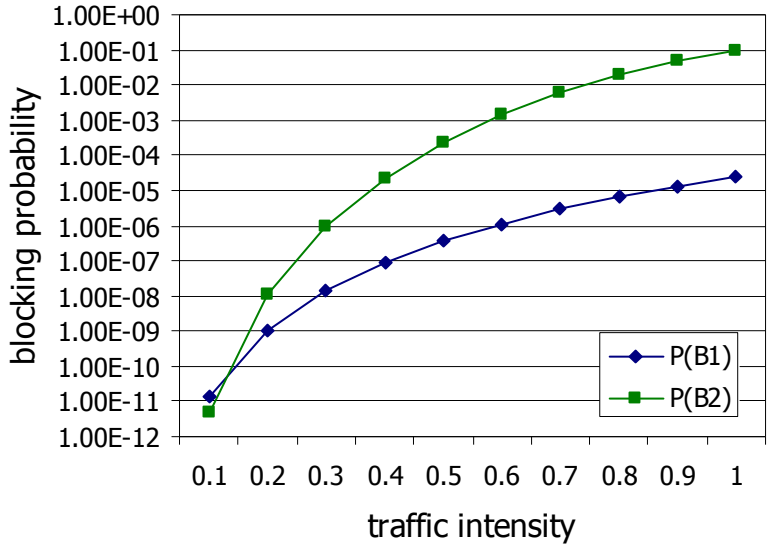


Figure 5-23: Blocking probability of a priority queue system with 2 priorities and 1 server

Figure 5-24 shows the waiting probability for different traffic intensities. It can be seen that with increasing traffic intensity, the probability that the server is busy increases and thus that jobs have to queue before being processed.

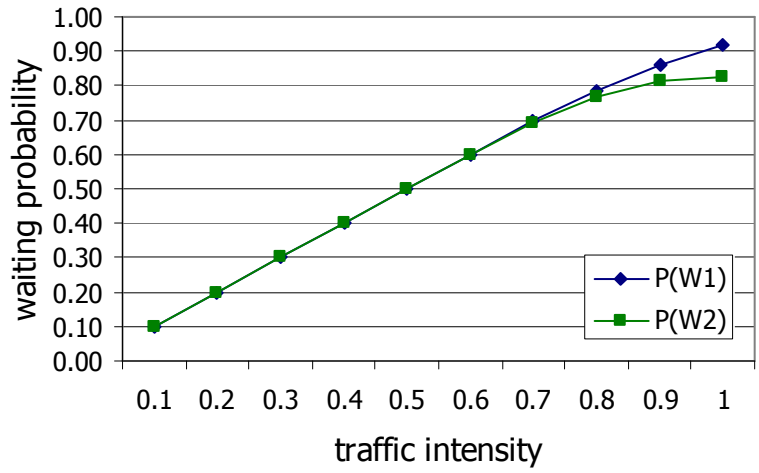


Figure 5-24: waiting probability of a priority queue system with 2 priorities and 1 server

Figure 5-25 shows that although the waiting probability increases with growing traffic intensity, the expected number of jobs in queue 1 does not increase significantly. This means that in situations with high traffic intensity, priority 1 jobs are first placed in the queue, but then taken out of the queue with high probability as soon as the server is empty.

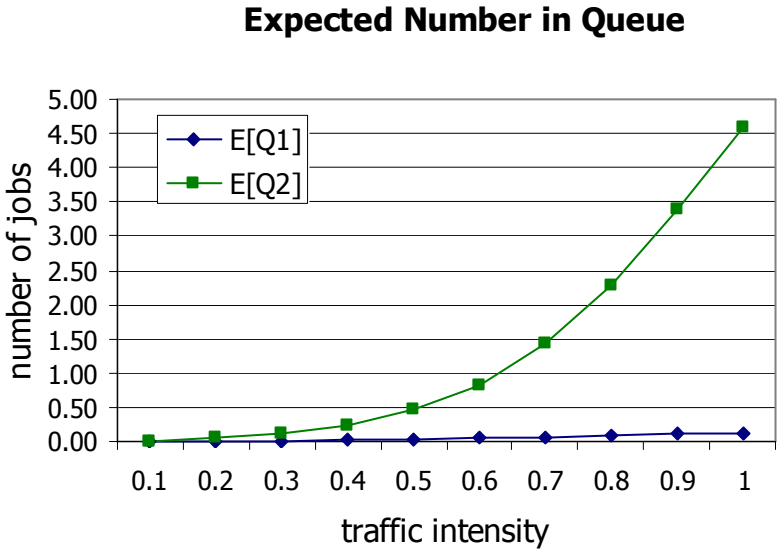


Figure 5-25: expected number in queues of a priority queue system with 2 priorities and 1 server

Table 5-4 shows the performance evaluation results for the same system, but with 2 server processes.

u	E[X]	E[Q1]	E[Q2]	T	Tprio[1]	Tprio[2]	P(B1)	P(B2)	P(W1)	P(W2)
0.1	0.2020	0.0002	0.0018	0.1000	0.0100	0.0900	2.4286E-12	8.4682E-13	0.0182	0.0182
0.2	0.4167	0.0015	0.0152	0.2000	0.0200	0.1800	3.4853E-10	3.5686E-09	0.0667	0.0667
0.3	0.6593	0.0047	0.0546	0.3000	0.0300	0.2700	6.3497E-09	4.3498E-07	0.1385	0.1385
0.4	0.9523	0.0108	0.1415	0.4000	0.0400	0.3600	4.9057E-08	1.2127E-05	0.2286	0.2286
0.5	1.3316	0.0204	0.3114	0.4999	0.0500	0.4499	2.3572E-07	1.4801E-04	0.3332	0.3331
0.6	1.8568	0.0341	0.6239	0.5994	0.0600	0.5394	8.3637E-07	1.0493E-03	0.4492	0.4482
0.7	2.6113	0.0523	1.1653	0.6969	0.0700	0.6269	2.3961E-06	4.9725E-03	0.5720	0.5671
0.8	3.6640	0.0745	2.0138	0.7878	0.0800	0.7078	5.8151E-06	1.6945E-02	0.6935	0.6765
0.9	4.9817	0.0994	3.1526	0.8649	0.0900	0.7749	1.2278E-05	4.3392E-02	0.8008	0.7574
1	6.3869	0.1245	4.4190	0.9217	0.1000	0.8217	2.3012E-05	8.7011E-02	0.8825	0.7955

Table 5-4: Performance measures of a priority queue system with 2 priorities and 2 servers

Table 5-5 shows the performance evaluation results for the same system with 3 server processes.

u	E[X]	E[Q1]	E[Q2]	T	Tprio[1]	Tprio[2]	P(B1)	P(B2)	P(W1)	P(W2)
0.1	0.3004	0.0000	0.0004	0.1000	0.0100	0.0900	4.9473E-13	1.7258E-13	0.0037	0.0037
0.2	0.6062	0.0005	0.0056	0.2000	0.0200	0.1800	1.2891E-10	1.3199E-09	0.0247	0.0247
0.3	0.9300	0.0024	0.0276	0.3000	0.0300	0.2700	3.2114E-09	2.2000E-07	0.0700	0.0700
0.4	1.2941	0.0067	0.0874	0.4000	0.0400	0.3600	3.0300E-08	7.4903E-06	0.1412	0.1412
0.5	1.7356	0.0145	0.2212	0.5000	0.0500	0.4500	1.6748E-07	1.0516E-04	0.2368	0.2367
0.6	2.3172	0.0269	0.4917	0.5996	0.0600	0.5396	6.5913E-07	8.2692E-04	0.3540	0.3532
0.7	3.1303	0.0446	0.9937	0.6973	0.0700	0.6273	2.0433E-06	4.2404E-03	0.4878	0.4836
0.8	4.2571	0.0675	1.8228	0.7890	0.0800	0.7090	5.2636E-06	1.5338E-02	0.6277	0.6124
0.9	5.6717	0.0939	2.9774	0.8668	0.0900	0.7768	1.1595E-05	4.0981E-02	0.7563	0.7154
1	7.1813	0.1208	4.2884	0.9240	0.1000	0.8240	2.2332E-05	8.4440E-02	0.8564	0.7720

Table 5-5: Performance measures of a priority queue system with 2 priorities and 3 servers

Figure 5-26 shows for priority 1 that the blocking probability in general does not depend very much on the number of servers in the system. This can be expected, as blocking depends more on the total mean system throughput than on the number of servers.

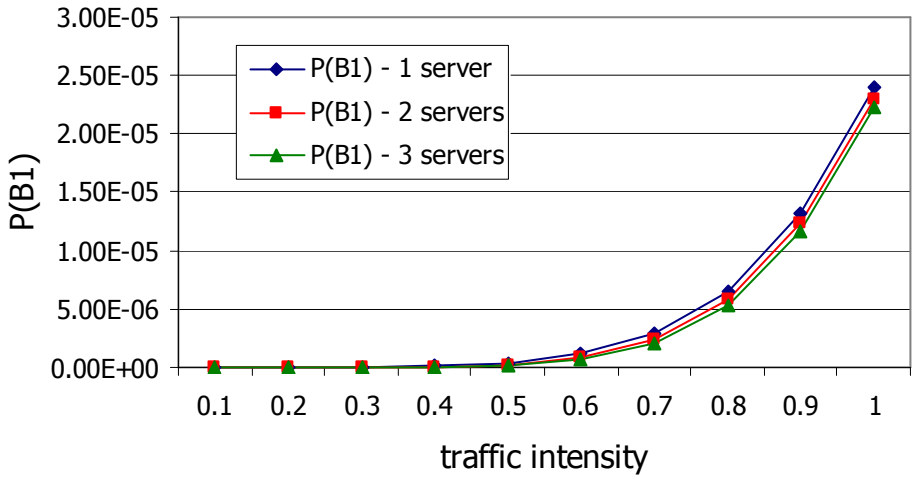


Figure 5-26: Blocking probability of priority 1 queue for different server scenarios

Similar to the blocking probability, the expected number of jobs in the queues does not depend heavily on the number of servers in the system (see Figure 5-27), but decreases slightly when increasing the number of server processes.

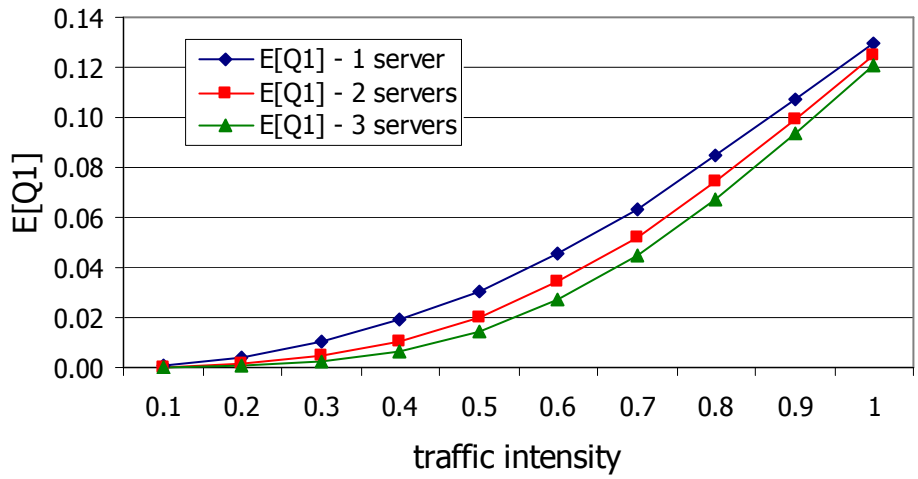


Figure 5-27: Excpected number of jobs in queue with priority 1 for different server scenarios

Figure 5-28 shows however that the waiting probability decreases significantly, when increasing the number of servers in the system. This can be explained intuitively that by increasing the number of servers, also the number of jobs that can be concurrently being processed by the system increase. Thus the probability for a job to arrive at a totally busy system decreases when increasing the number of servers.

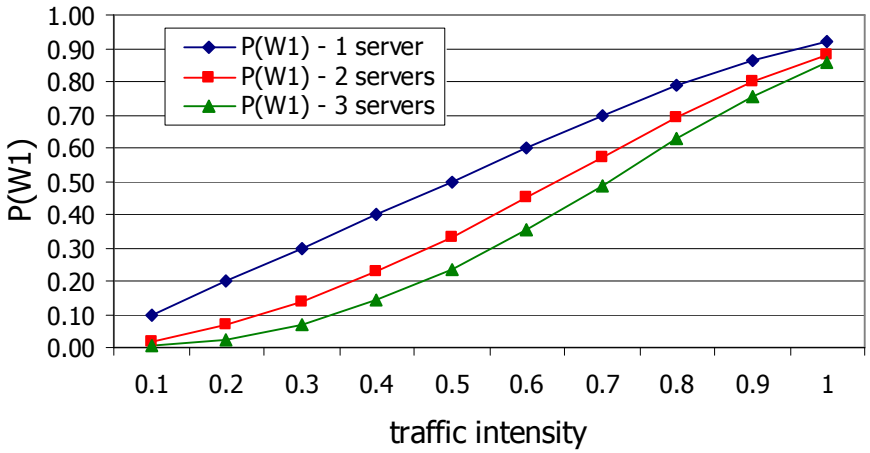


Figure 5-28: Waiting probability of priority 1 queue for different server scenarios

These results show that the maximum arrival rates and queue lengths for given blocking probabilities can be approximated by simulating a multi threaded system by a single server. It is especially important that the performance values for blocking probabilities, waiting probabilities and expected queue lengths of a single server system are upper bounds for a system with more parallel servers.

It would nevertheless be interesting to calculate the performance values for a system that e.g., has 50 parallel server threads (m = 50), a value that a real application server should be able to handle. Such a large number of server processes however increases the number of system states by a factor of $(n + 1)^m$ to a magnitude that cannot be calculated on the computers that were available for this thesis.

5.4 Mobility Simulation for a Terminal-Based Location Enabler

This section investigates the traffic source characteristics of an IMS Location Presence system by means of a mobility model. First a simulation is used to compare the number of triggered location updates to the number of requests in a polling system in order to estimate the gains in access network capacity for the proposed terminal-based location enabler architecture. Second the simulation is used to analyse the statistical parameters of the simulated traffic.

5.4.1 Access Network Capacity Gains

A network-based positioning method is forced to implement triggered location updates through polling. There is no indication in the network that a mobile terminal changes its position as long as it remains in the same cell or even in the same location area (a number of cells controlled by one Mobile Switching Centre). Furthermore mobile terminals tend to stay in an idle state most of the time in order to conserve battery power. Hence position changes are not visible to the network. This means that a tracing application has to poll for the location of a terminal periodically, even if the movement of the observed node does not require a new position fix.

A terminal based location-enabler however is able to implement trigger logic directly in the terminal so that necessary location updates are kept to an absolute minimum and scarce resources in the radio access network are used in the most economic way.

A simulation based performance analysis of a terminal-based IMS location enabler was done for this thesis in order to investigate the quantitative advantage of triggered location updates of a terminal based location enabler compared to the polling of a network based location enabler. The simulation of node movements is based on the Random Waypoint Model that was first introduced by J. Broch et al. in [Broch] and is widely used as a mobility model to compare the performance of various mobile network protocols. J. Yoon et al. show in [Yoon] that a simulation based on the Random Waypoint Model will only reach a steady state, if a minimum speed is defined in the simulation.

In the Random Waypoint Model (RWM) a node moves from its current location to a new location by randomly choosing a new location in the simulation area and a speed that is uniformly distributed between [*minspeed, maxspeed*]. The RWM includes randomly chosen pause times between changes in direction and/or speed.

The BonnMotion simulation software was used for the mobility simulation. BonnMotion [BonnMotion] is a mobility scenario generation and analysis tool developed within the Communication Systems group at the Institute of Computer Science IV of the University of Bonn, Germany. The simulation targets a group of 100 pedestrians that move in a square with an edge length of 500 m. The node speed is chosen between [0.2, 1.5] m/sec. The maximum pause time is 60 sec. The simulation was run for 7200 sec in order to reach a steady state node distribution and then simulated measurements were taken for 1800 sec. Figure 5-29 exemplarily shows the movement traces of 5 nodes.

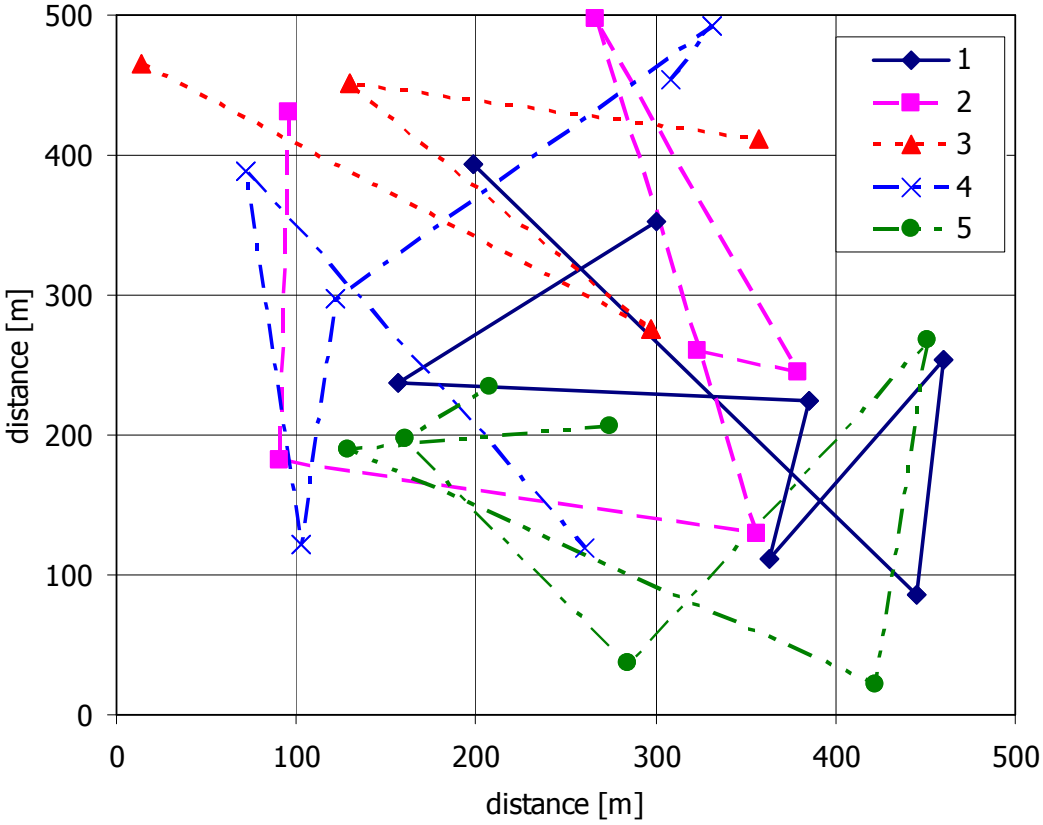


Figure 5-29: Node movement in the Random Waypoint Model

The simulation set-up was chosen in a way to estimate a lower bound on the reduction of necessary location update messages in a triggered scheme compared to a polling scheme. Our simulation observes nodes that are nearly constantly moving. Permanent movement is the worst case scenario for the number of triggered location updates, because location update messages have to be sent in regular intervals. In a real world scenario, people do not move constantly but rather tend to stay in one place for long periods of time (e.g., commuters between home and office). As a triggered location update is not fired when a node does not move, it can be expected that savings in the number of necessary location update messages and thus in scarce wireless network capacity is much higher in real world movement patterns.
The node movement traces that come out of the RWM simulation were analyzed in two ways:

- Count the number of triggered location updates that are necessary to report location changes for a given spatial resolution in meters.
  The actual position was checked every second and if the distance of the actual position to the last reported position was greater than the given spatial resolution a location updated was triggered. This implies that the maximum error in the location update is the distance that the node can move at maximum speed in one second. For a high spatial resolution of 10 m and a maximum speed of 1.5 m/sec the error results in 15% at most.
- Compute the optimal polling time and the resulting number of location requests.
  The optimal polling time is the interval between sending location requests to the network that result in a minimal absolute spatial error of all reported positions. Absolute spatial error indicates a deviation from the required spatial resolution in both directions – meaning that the polling happens either too often or too seldom.

Figure 5-30 shows the optimal polling time at 145 sec where the relative location error is lowest for a scenario with a spatial resolution of 10m. Even at that minimum, the relative location error is still 43%. In Figure 5-31 the required number of samples is given depending on the polling time. A polling time of 145 sec gives a number of 1241 request during the simulation run. The number of triggered updates during the simulation was counted to be 863.

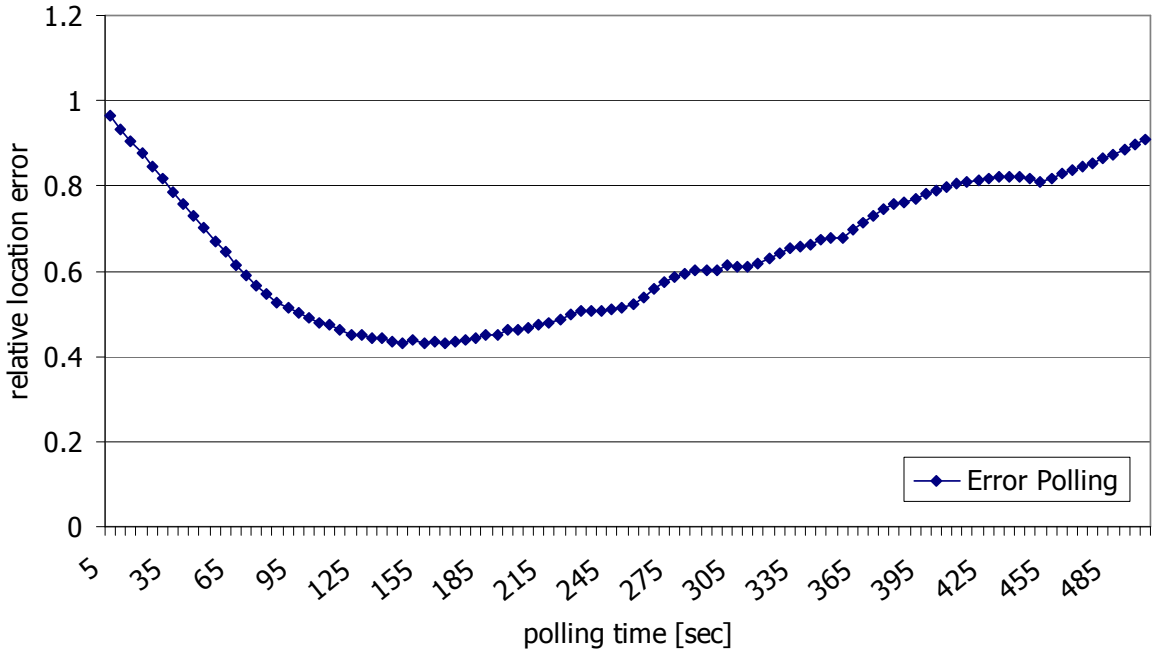


Figure 5-30: relative location error of a polling scheme for a spatial resolution of 10m

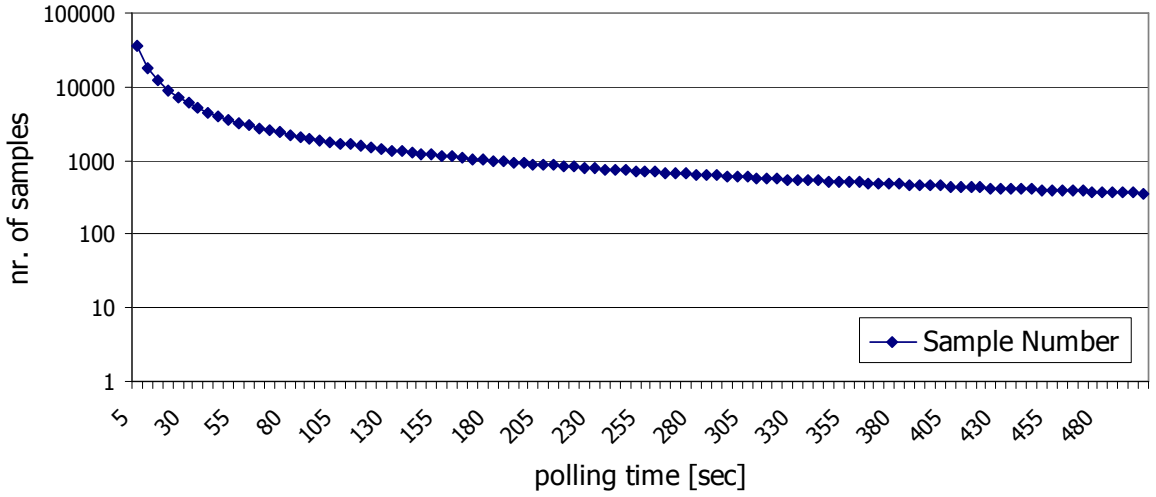


Figure 5-31: number of samples depending on the polling time for a scenario with a spatial resolution of 10m.

The optimal polling time is a function of the required spatial resolution and of the movement pattern and is therefore difficult to find for real applications. Even when choosing the optimal polling time, the average spatial error of all reported positions was about 40% compared to the required spatial resolution.

The number of triggered location updates was compared to the number of location requests at the optimal polling time for different values of the spatial resolution (see Figure 5-32). This comparison shows that a triggered location update scheme can save at least 25% of messages, compared to a polling scheme working at the optimal polling time. Furthermore node locations in a triggered scheme are reported with at least half the error than for a polling scheme.

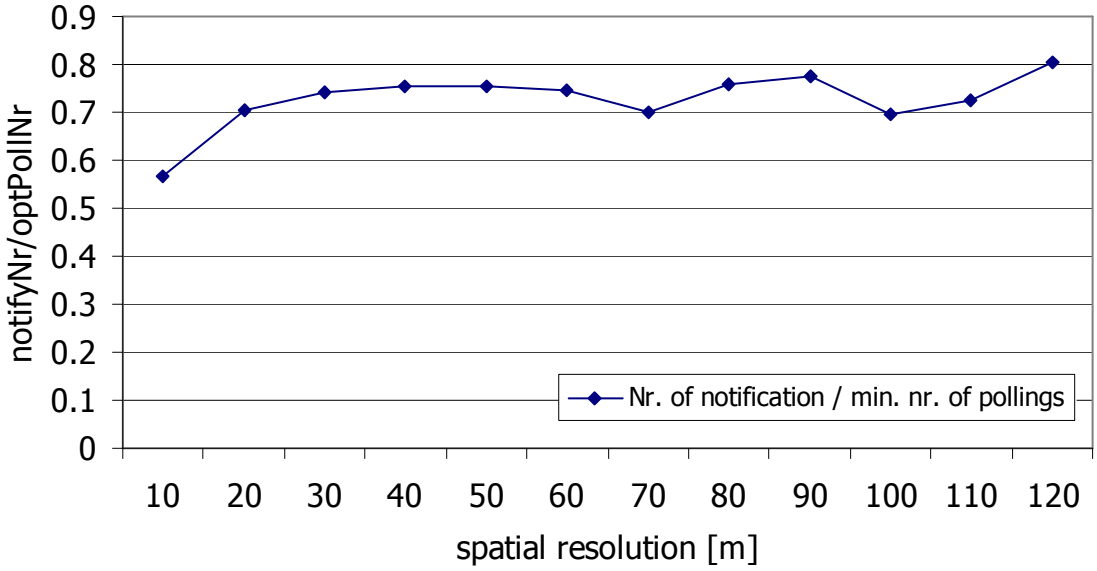


Figure 5-32: Number of triggered location updates compared to the number of location requests at the optimal polling time

These performance values represent a lower bound to the possible savings, as the optimal polling time is difficult to find in real applications. Furthermore real world movement patterns will require much less location updates than the set of constantly moving nodes in the simulation.

A terminal based triggered location update scheme can achieve the highest savings, if only a small set of location events are of interest to an application. The profile of a mobile phone could for example be switched depending on the location events 'at home', 'in the office' and 'on the road'. It is clear that implementing such an application by polling a network based location enabler is not feasible for networks with millions of subscribers.

5.4.2 Statistical Parameters of a Simulated Source

The simulation of location update notifications based on the Random Waypoint model was also used to analyse the statistical parameters of the resulting traffic.

The simulation was run for different spatial resolutions of the tracing application and the interarrival times of position update notifications were calculated for all updates. The simulation parameters were:

- 100 nodes
- x = 500 m
- y = 500 m
- duration = 1800 sec
- v_{min} = 0.2 m/sec
- v_{max} = 1.5 m/sec
- t_{pause_max} = 60 sec

Figure 5-33 shows that the mean inter arrival time of location update notifications (see also Table 5-6) has a quadratic dependency to the spatial respolution:

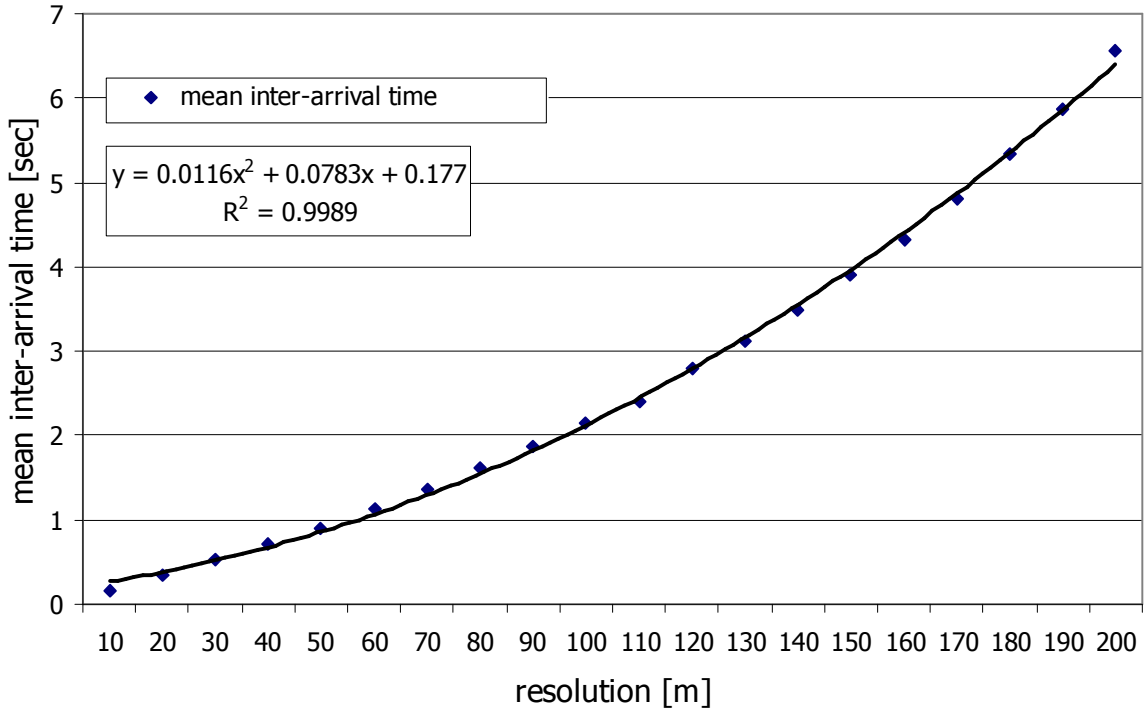


Figure 5-33: Mean inter-arrival time over spatial resolution for a Random Waypoint Model

Spatial resolution	Mean inter arrival time
10	0.1674388
20	0.33698906
30	0.52061577
40	0.70817134
50	0.91002943
60	1.12815017
70	1.36335164
80	1.60785512
90	1.87484195
100	2.14702405
110	2.40889232
120	2.784487
130	3.11902357
140	3.4993432
150	3.89463431
160	4.32851999
170	4.81660115
180	5.33586379
190	5.87589513
200	6.55269592

Table 5-6: mean arrival rates for different spatial resolutions

Running the simulation for different numbers (10, 50, 100, 200) of nodes gives the mean inter arrival times T_i. The mean arrival rate $\lambda = 1/ T_i$ shows a linear dependency from the number of nodes (see Figure 5-34).

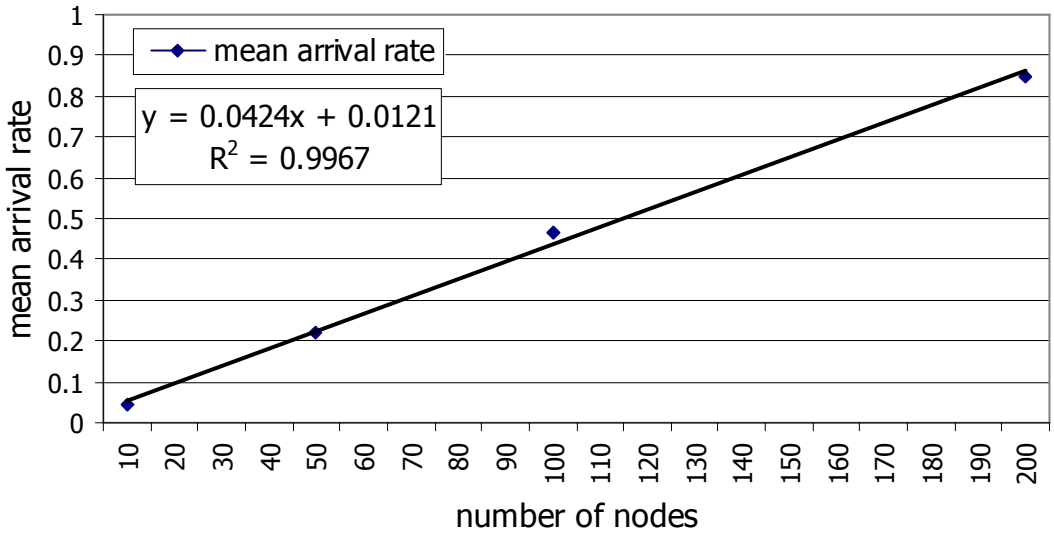


Figure 5-34: mean arrival rate over number of nodes

Each run of the node movement simulation will result in slightly different values for the mean interarrival time. The first moment of the interarrival time shall be regarded as a population parameter, and in the following the 95% confidence interval is calculated.

A confidence interval for a population parameter is an interval with an associated probability p that is generated from a random sample of an underlying population such that if the sampling was repeated numerous times and the confidence interval recalculated from each sample according to the same method, a proportion p of the confidence intervals would contain the population parameter in question.

It is suppsed that $X_1, X_2,\ldots,X_n$ are independent samples of a normally distributed population with the mean μ and the variance σ^2. The mean and the variance are estimated from the samples by

$$\overline{X} = \frac{(X_1 + X_2 + \cdots + X_n)}{n}$$

$$S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X})^2$$

The random variable T

$$T = \frac{\overline{X} - \mu}{S/\sqrt{n}}$$

has a Student-t distribution with n-1 degrees of freedom (Note that the distribution of T does not depend upon the unobservable parameters μ and σ^2).

A 95% confidence interval means that

$$1 - \alpha = 0.95$$

$$P(-t(1-\tfrac{\alpha}{2}, n-1) < T < t(1-\tfrac{\alpha}{2}, n-1)) = 1-\alpha$$

where $t(1-\alpha/2)$ is the $1-\alpha$ quantile of the Student-t distribution.

Consequently the confidence interval for μ follows with:

$$P(\overline{X} - t(1-\tfrac{\alpha}{2}, n-1)\frac{S}{\sqrt{n}} < \mu < \overline{X} + t(1-\tfrac{\alpha}{2}, n-1)\frac{S}{\sqrt{n}}) = 1-\alpha$$

For 10 runs of the simulation for a spatial resolution of 100 m the mean interarrival times in are observed.

Mean inte-rarrival time
2.14702405
2.33199101
2.26032807
2.24121433
2.25003512
2.35242629
2.32130977
2.33536088
2.22882657
2.21782888

Table 5-7: mean interarrival times for different simulation runs (R = 100 m)

From these observed interarrival times, the following 95% confidence interval in calculated:

$2.24 < \mu < 2.3$

The mean inter arrival time can be used for a basic network and server capacity planning. A tracking application for a spatial resolution of 20 m will have a mean arrival rate of 1/0.336989/100 = 0.02967 notifications per second and per user. For 10000 concurrent users, this results in a total traffic of about 300 requests/sec at the location presence server. Estimating the message size of a single notification to be about 1.5 kByte, the location presence server will need a core network capacity of 3.5 Mbit/sec.

Form the simulated node movements the histograms and statistical parameters of the interarrival times for different spatial resolutions were calculated.

For a spatial resolution R of 10m the following parameters are found by simulation:

- Interarrival time mean: 0.167 [sec]
- Interarrival time std. deviation σ_t: 0.205 [sec]
- mean arrival rate λ 5.97 [1/sec]

Figure 5-35 shows a histogram of the interarrival times for the simulation with a spatial resolution of R=10 m. The observed values are compared to calculated values for a negative-exponential distribution.

The bin width for the histogram is chosen to be about 0.3 times the standard deviation σ_t of the observed interarrival times. The bin b_i contains the observed number of arrivals A_i that happen in the time interval $t_i<t<t_{i+1}$ after the last arrival where

$$t_{i+1} - t_i \approx 0.3\sigma_t$$

The observed number A_i is compared with the calculated number of arrivals B_i from a negative exponential distribution.

$$B(t) = 1 - e^{-\lambda t}$$

$$B_i = B(t_{i+1}) - B(t_i)$$

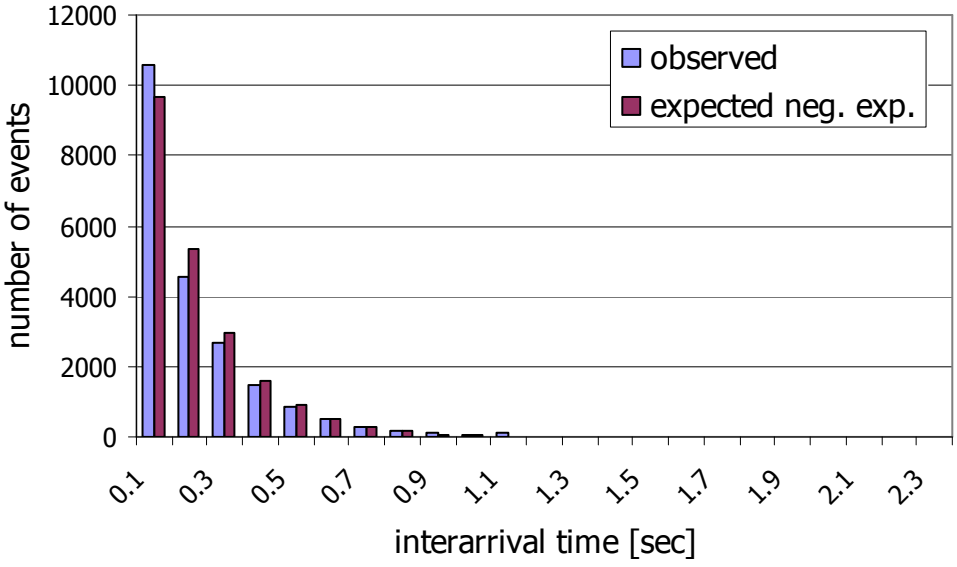


Figure 5-35: histogram of simulated interarrival times for resolution R=10 versus a negative-exponential distribution

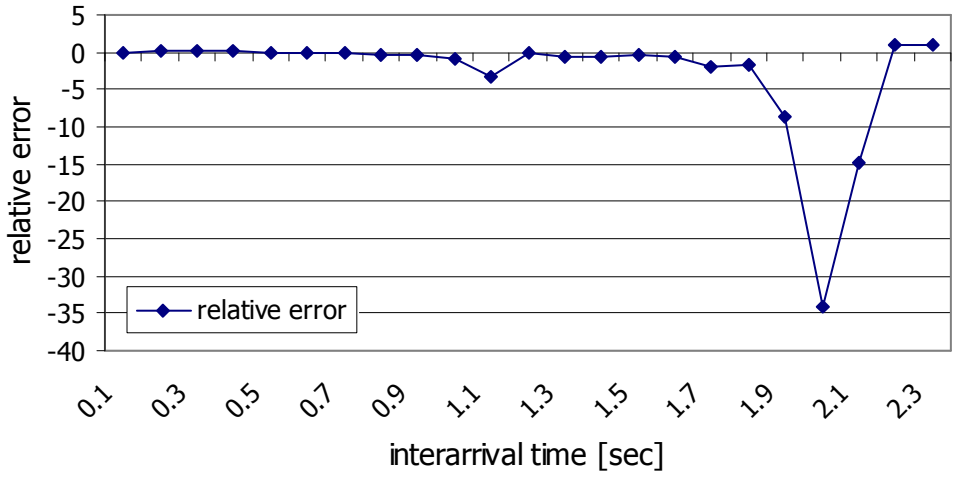


Figure 5-36: errors of simulated interarrival times for resolution R=10 relative to a negative-exponential distribution

Figure 5-36 shows the relative error of the observed interarrival times to the values from the negative exponential distribution.

$$E_i^{(B)} = \frac{B_i - A_i}{B_i}$$

Form the charts above, it can be seen that the simulated interarrival times deviate only for large values of the interarrival time from the negative exponential distribution.
For a spatial resolution R of 100 m the following parameters are found by simulation:

- Interarrival time mean: 2.147 [sec]
- Interarrival time std. deviation σ_t: 3.078 [sec]
- mean arrival rate λ: 0.47 [1/sec]

For a spatial resolution R of 200 m the following parameters are found by simulation:

- Interarrival time mean: 6.553 [sec]
- Interarrival time std. deviation σ_t: 9.898 [sec]
- mean arrival rate λ: 0.15 [1/sec]

Figure 5-37 shows the histogram of the interarrival times for a spatial resolution of R = 100 m and Figure 5-38 shows the errors relative to the calculated values from a negative exponential distribution. Figure 5-39 and Figure 5-40 show the same for a spatial resolution of R = 200 m. It can be seen that the deviation from the calculated values decreases with increasing spatial resolution, but that the traffic source cannot be modelled satisfactorily by one parameter distributions like a negative exponential distribution or an Erlang-2 distribution.

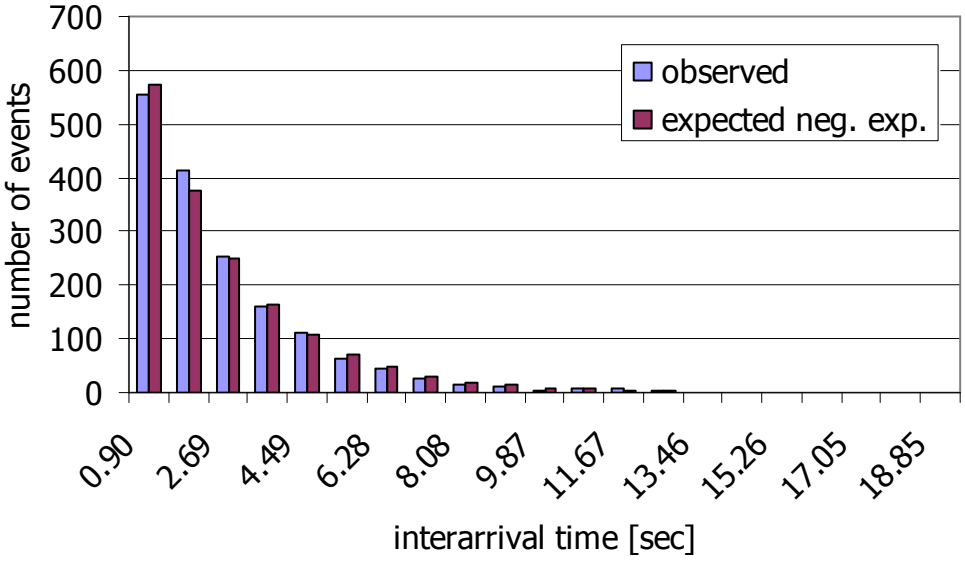


Figure 5-37: histogram of simulated interarrival times for resolution R=100 versus a negative-exponential distribution

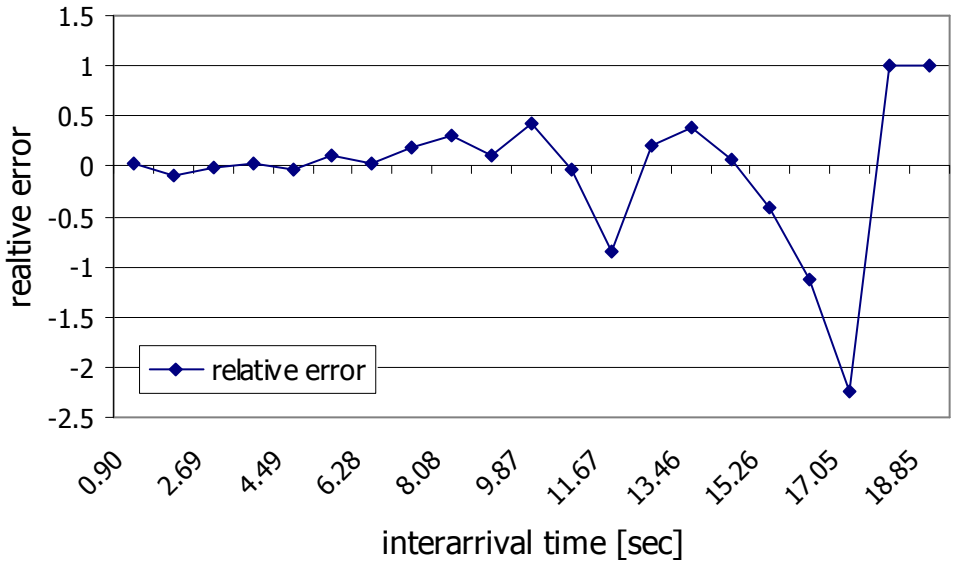


Figure 5-38: errors of simulated interarrival times for resolution R=100 relative to a negative-exponential distribution

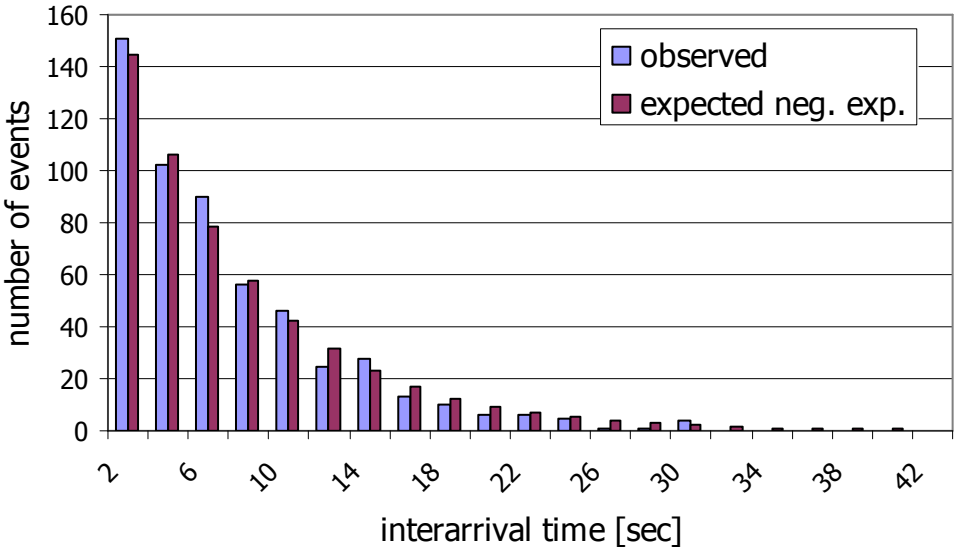


Figure 5-39: histogram of simulated interarrival times for resolution R=200 versus a negative-exponential distribution

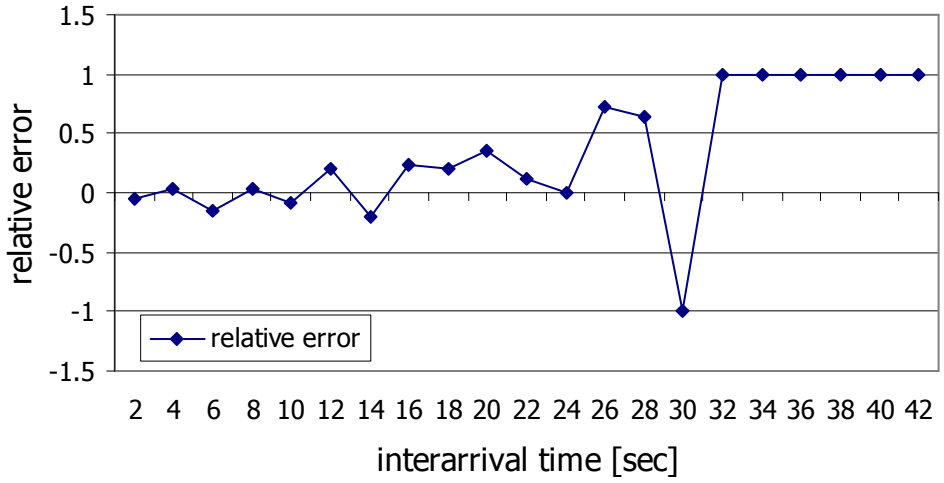


Figure 5-40: errors of simulated interarrival times for resolution R=200 relative to a negative-exponential distribution

From the histogram of interarrival times for a spatial resolution R = 200 (Figure 5-39) it can be seen, that the simulated data is similar to the calculated values of a negative exponential distribution. A Chi-square test was therefore performed to test the hypothesis, that the observed data fits a negative exponential distribution with a given significance level.

A chi-square test is used to test, if a sample of data comes from a population with a specific distribution. The hypothesis that gets tested is that the data follows the specified distribution.

For the chi-square goodness-of-fit computation (see [Kreyszig] for more details) the data is divided into k intervals such that each interval contains at least 5 values of the given samples.

For the interval i the observed values lie between the upper limit U_i and the lower limit L_i. The expected frequency E_i in the interval for the specified cumulative distribution function F(x) and the sample size N is:

$$E_i = N\left(F(U_i) - F(L_i)\right)$$

The deviation χ^2 is computed with the observed frequency O_i

$$\chi_0^2 = \sum_{j=1}^{k} \frac{(O_i - E_i)^2}{E_i}$$

The critical value c is found for a given significance level of α by solving the equation

$$P(\chi^2 \le c) = 1 - \alpha$$

The hypothesis that the data follows a given distribution is rejected if

$$\chi_0^2 > c$$

The critical value c can be found in tables of the chi-square distribution with k-1 degrees of freedom.

In the histogram of the interarrival times for R = 200, there are 12 intervals that have more than 5 entries, thus giving k=12 (see Table 5-8). The chi-square value is $\chi_0^2 = 44{,}1$.

interarrival time	Neg. exponential	observed	chi-square of interval
2	144.41	151	0.3
4	106.42	102	0.18
6	78.43	90	1.89
8	57.80	56	1.95
10	42.60	46	2.22
12	31.39	25	3.52
14	23.13	28	4.54
16	17.05	13	5.51
18	12.56	10	6.03
20	9.26	6	7.18
22	6.82	6	7.28
∞	19.21	11	3.51
chi-square			44.10

Table 5-8: Chi-square value for a spatial resolution of 200 m

From the chi-square distribution table, the critical value c for a significance value of α=5% and K = 12 can be found to be c = 21,03.

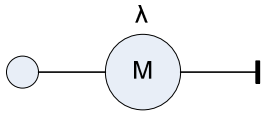
This means that the hypothesis that the observed data comes from a negative exponential distributed population has to be rejected. The histogram of the data observed in the simulation has a longer tail, than a negative exponentially distributed data set. It can be seen in Table 5-8 that the number of observed events until an interarrival time of 16 seconds would not be rejected by the hypothesis tests. Only the last three intervals let the chi-square value increase substantially.

This result means that other methods must be employed to find a model that fits the probability distribution of the simulated data. One such method is proposed by T. Osogami and M. Harchol-Balter in [Osogami] that fits the first three moments of a given general distribution to a phase-type distribution. This method is presented in the next section.

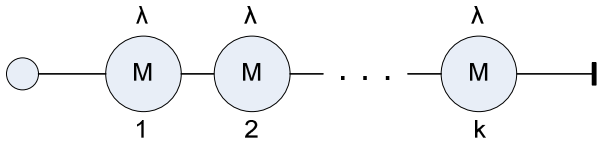
5.5 Fitting of Measured or Simulated Data by Phase-Type Distributions

One of the most commonly used method for the exact or approximate analysis of queuing systems is the approximation of generally distributed interarrival or service times by a phase-type (PH) distribution. A phase-type distribution is a probability distribution that results from connecting several inter-related Poisson processes in phases. The distribution can be represented by a random variable describing the time until absorption of a Markov Chain with one absorbing state. Each of the states of the Markov Chain represents on of the phases.
The following probability distributions are considered special cases of a phase-type distribution:

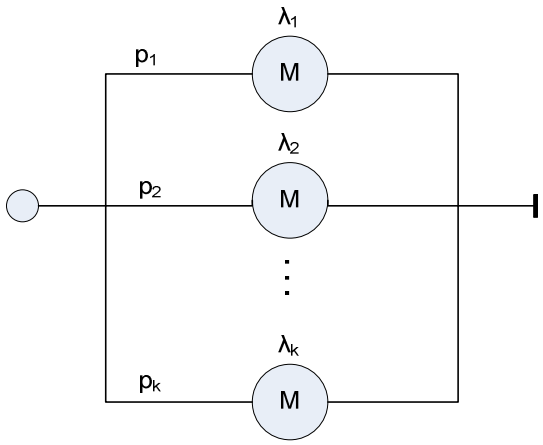
- Negative-exponential distribution – 1 phase (coefficient of variation C = 1)



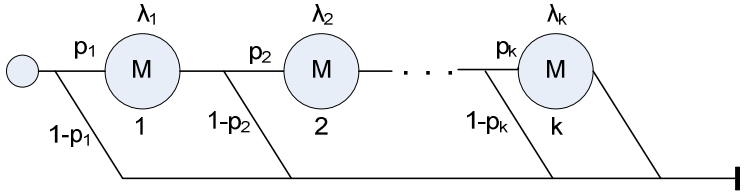
- Erlang distribution – 2 or more identical phases in sequence (C < 1)



- Hyperexponential distribution - 2 or more non-identical phases, that each have a probability of occurring in a mutually exclusive, or parallel, manner. (C > 1)



- Coxian distribution - 2 or more (not necessarily identical) phases in sequence, with a probability of transitioning to the terminating/absorbing state after each phase



Due to the memoryless property of the negative-exponential distribution, the stochastic processes are of Markovian type at the phase level so that well known methods of analysis can be applied.

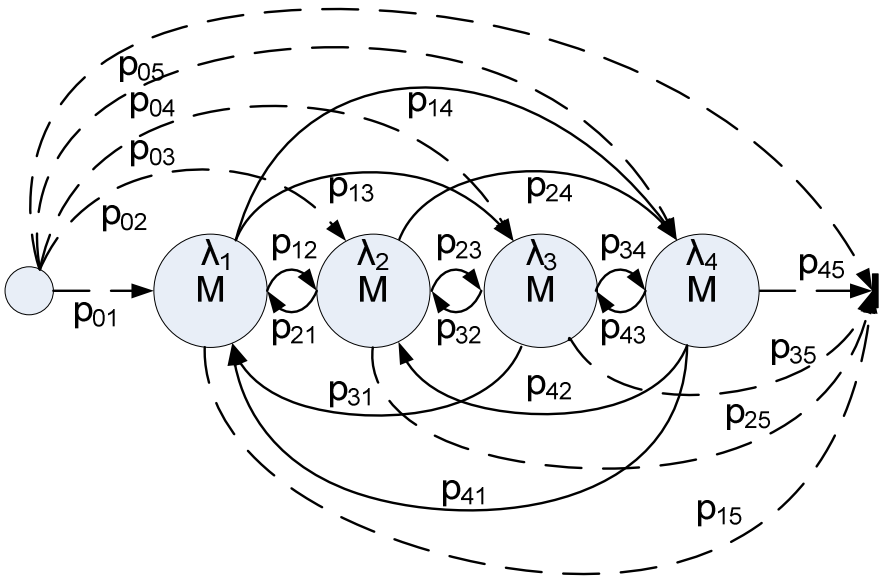


Figure 5-41: A 4 phase PH distribution

Figure 5-41 shows a 4 phase PH distribution. There a n = 4 phases, where the i-th state has negative-exponentially distributed service time with rate λ_i. With probability p_{0i} the process starts in the i-th state. The transition probability from state i to state j is p_{ij}. Each state has a probability p_{i5} that it is the last state.

5.5.1 The Osogami, Harchol-Balter Approximation Method

In this section an approximation method is first presented and then applied that has been proposed by T. Osogami and M. Harchol-Balter in the paper "A closed-form solution for mapping general distributions to minimal PH distributions" [Osogami]. There an algorithm is proposed for mapping a general distribution G to a PH distribution where the goal is to find a PH distribution which matches the first three moments of G. G is called *well represented* by PH, if the first three moments agree. This algorithm will henceforward be called OHB-method. The OHB method provides a closed form representation of the parameters of the matching PH distribution, applies to all distributions which can be well-represented by a PH distribution, and is nearly minimal in the number of phases required.

The following presents the algorithm of the OHB method and is an excerpt from [Osogami].

The first step in finding a moment matching algorithm is to reduce the number free parameters by only considering a subset S of all possible PH distributions. From Figure 5-41 it can be seen that a general PH distribution has $O(n^2)$ free parameters. S has to be chosen with care. If S is too small this may limit the space of distributions which are well-represented, or may exclude solutions with minimal number of phases.

The OHB method defines a subset S of all PH distributions that is called EC distributions. EC distributions only have 6 free parameters, which allows to develop a closed-form solution for the fitting to a general distribution G. [Osogami] proves that for all distributions G that can be well-represented by a PH distribution there exists an EC distribution, E, such that G is well-represented by G.

The OHB method works with the normalized k-th moment of a distribution.

Let $E[X^k]$ be the k-th moment of a distribution X for k = 1,2,3

The normalized k-th moment m_k^X of X for k=2,3 is then defined as:

$$m_2^X = \frac{E[X^2]}{(E[X])^2} \quad and \quad m_3^X = \frac{E[X^3]}{E[X]E[X^2]}$$

Note the correspondence to the coefficient of variation C (standard deviation σ):

$$C = \frac{\sigma}{E[X]}$$

$$C^2 = \frac{E[X^2] - (E[X])^2}{(E[X])^2}$$

$$m_2^X = C^2 + 1$$

Also note the correspondence to the skewness γ:

$$m_3^X = \gamma\sqrt{m_2^X}$$

It can be shown that all distributions G can be well-represented by PH distributions, if

$$m_3^G > m_2^G > 1$$

which holds for almost all nonnegative distributions G.

The OHB method defines OPT(G) to be the minimum number of necessary phases for a distribution G to be well-represented by an acyclic PH distribution, where an acyclic PH distribution is a PH distribution in which there is no transition from state i to state j for all $i > j^2$.

The EC distribution is defined as follows:

A random variable for the time until absorption follows an EC distribution if with probability 1-p, the value is zero, and with probability p, the value is an Erlang-(n-2) distribution followed by a two-phase Coxian distribution for integers $n \geq 2$ (see Figure 5-42).

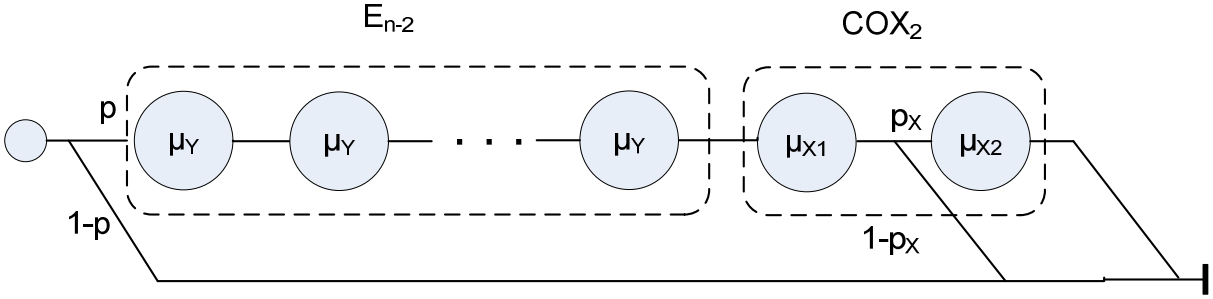


Figure 5-42 An EC distribution

A Coxian distribution is very good for approximating any distribution with high variability. In particular, a two-phase Coxian distribution is known to well-represent any distribution G that has:

$$m_2^G > 2 \quad and \quad m_3^G > \frac{3}{2} m_2^G$$

However a Coxian distribution requires many more phases for approximating distributions with lower second and third moments (e.g., a Coxian distribution requires at least n phases to well-represent a distribution G with

$$m_2^G \leq \frac{n+1}{n} \quad for \quad n \geq 1$$

By contrast, an Erlang distribution has only two free parameter and is also known to have the least normalized second moment among all the PH distributions with a fixed number of phases. However the Erlang distribution is obviously limited in the set of distributions which it can well-represent.

The OHB method thus combines an Erlang-k distribution with a two-phase Coxian distribution that allows to represent distributions with all ranges of variability, while using only a small number of phases.

The EC distribution has the 6 free parameters

$$n, p, \mu_Y, \mu_{X1}, \mu_{X2}, p_X$$

that must be found in order to represent a given distribution G.

For finding the parameters of the EC distribution that maps a given distribution G in the first three moments, the following reasoning is given. If the distribution G has sufficiently high second and third moments, then a two-phase Coxian distribution alone suffices and zero phases of the Erlang distribution are needed. If the variability of G is lower, however, a number of Erlang phases are added in order to get the variability low enough. The Erlang phases are chosen so that the overall variability is minimized.

The OHB method starts by defining an operation A(X) on an arbitrary distribution X such that

$$A(X) = Y + X$$

where Y is a negative exponential distribution independent of X, and the mean of Y is chosen so that the normalized second moment of A(X) is minimized.

Applying operation A several times, which means that additional negative exponential phases are added gives the operation $A^m(X)$:

$$A^m(X) = A(A^{m-1}(X)) \quad for \quad m \geq 1 \quad and \quad A^0(X) = X$$

In [Osogami] the proof can be found that if the exponential distribution Y being appended by operation A is chosen so as to minimize the normalized second moment of A(X) (as specified by the definition), then the mean of each successive Y is always the same and is defined by

$$A^m(X) = Y_m + A^{m-1}(X) \quad for \quad m = 1, \ldots, n$$

$$\frac{1}{\mu_Y} = E[Y_m] = (m_2^X - 1)E[X] \tag{1}$$

The normalized moments of $Z=A^n(X)$ are

$$m_2^Z = \frac{(m_2^X - 1)(n+1) + 1}{(m_2^X - 1)n + 1}$$

$$m_3^Z = \frac{m_2^X m_3^X + (m_2^X - 1)n\left(3m_2^X + (m_2^X - 1)(m_2^X + 2)(n+1) + (m_2^X - 1)^2(n+1)^2\right)}{\left((m_2^X - 1)(n+1) + 1\right)\left((m_2^X - 1)n + 1\right)^2} \tag{2}$$

The number of times that the operation A has to be applied to X in order to bring m_2^Z into the desired range, given the value m_2^X is limited by the following inequality (proof see [Osogami]). Note, that by choosing X to be a 2-phase Coxian distribution, m_2^X can only take on values greater than 2.

$$Z = A^n(X)$$

$$If \quad X \in \left\{F \mid 2 < m_2^F\right\}, \tag{3}$$

$$then \quad Z \in \left\{F \mid \frac{n+2}{n+1} < m_2^F < \frac{n+1}{n}\right\}$$

From the formulas above it can be seen that the mean μ_Y of the EC distribution depends on the remaining free parameters (n, p, μ_{x1}, μ_{x2}, p_x) that will determine E[X] and m_2^X. At least three free degrees of freedom are necessary to match three moments. The additional degrees of freedom are used in the OHB method to achieve a small number of phases and numerical stability.

The closed form for solution of the OHB method is based on the following classification of distributions. Let U_1, U_2, M_1, M_2 and L be sets of distributions defined as follows:

$$U_1 = \{F \mid m_2^F > 2 \text{ and } m_3^F > 2m_2^F - 1\},$$

$$U_2 = \{F \mid 1 < m_2^F < 2 \text{ and } m_3^F > 2m_2^F - 1\},$$

$$U = U_1 \cup U_2,$$

$$M_1 = \{F \mid m_2^F \geq 2 \text{ and } m_3^F = 2m_2^F - 1\},$$

$$M_2 = \{F \mid 1 < m_2^F < 2 \text{ and } m_3^F = 2m_2^F - 1\},$$

$$M = M_1 \cup M_2,$$

$$L = \{F \mid m_2^F > 1 \text{ and } m_2^F < m_3^F < 2m_2^F - 1\}$$

Figure 5-43 shows the classification sets in graphical from.

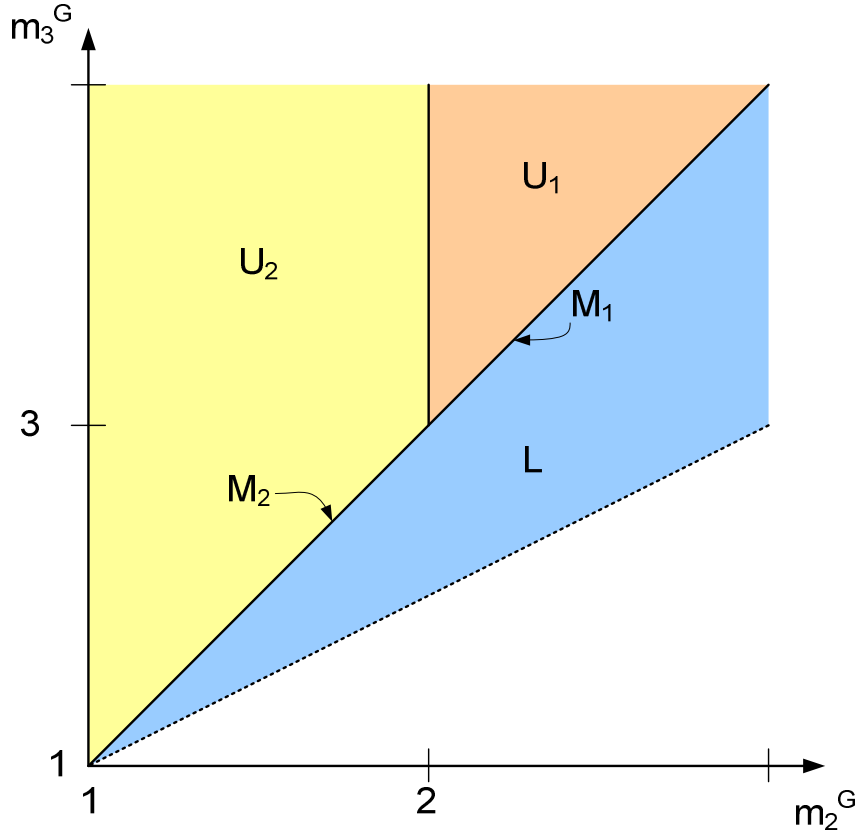


Figure 5-43: A classification of distributions. The dotted lines delineate the set of all nonnegative distributions $G(m_3^G \geq m_2^G \geq 1)$

In [Osogami] the proof can be found that for all distributions X, the distributions X and A(X) lie in the same classification region.

5.5.1.1 Simple Closed-Form Solution

The OHB method provides a simple closed-form solution for all distributions G that can be well-represented by a PH distribution and comply to the following characterization:

$$G : G \in \Phi,$$

$$\Phi = \left(U \cap \left\{ F \mid m_2^F \neq \frac{n+1}{n} \text{ for } n \geq 1 \right\} \right) \cup \left((M \cup L) \cap \left\{ F \mid m_2^F \neq \frac{n+1}{n+2} m_3^F \text{ for } n \geq 1 \right\} \right)$$

Observe, that Φ includes almost all distributions that can be well-represented by a PH distribution.

The solution differs according to the classification of the input distribution G.

a) $G \in U_1 \cup M_1$

A two-phase Coxian distribution suffices to match the first three moments of G:

$n = 2,$

$p = 1,$

$$\mu_{X1} = \frac{u + \sqrt{u^2 - 4v}}{2E[G]}$$

$$\mu_{X2} = \frac{u - \sqrt{u^2 - 4v}}{2E[G]}$$

$$p_X = \frac{\mu_{X2} E[G](\mu_{X1} E[G] - 1)}{\mu_{X1} E[G]}$$

$$where \quad u = \frac{6 - 2m_3^G}{3m_2^G - 2m_3^G} \quad and \quad v = \frac{12 - 6m_2^G}{m_2^G (3m_2^G - 2m_3^G)}$$

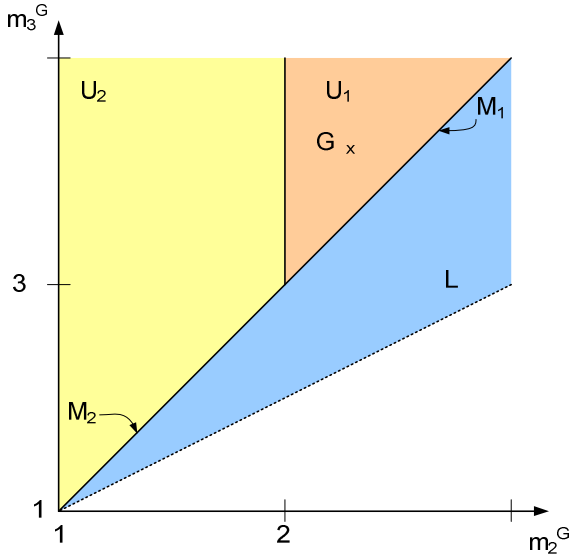


Figure 5-44: G is well represented by a 2-phase Coxian distribution

b) $G \in U_2 \cup M_2$

The minimum number of necessary phases follows from inequality (3).

$$n = \min\left\{k \mid m_2^G > \frac{k}{k-1}\right\} = \left\lfloor \frac{m_2^G}{m_2^G - 1} + 1 \right\rfloor \qquad (4)$$

Note: The floor function of a real number x, denoted $\lfloor x \rfloor$ or floor(x), is a function that returns the largest integer less than or equal to x.

Next the 2-phase Coxian distribution X has to be found so that G is well represented by $Z=A^n(X)$ according to equation (1). This is done by solving the equations in (2):

$$m_2^X = \frac{(n-3)m_2^G - (n-2)}{(n-2)m_2^G - (n-1)}$$

$$m_3^X = \frac{\alpha m_3^G - \beta}{m_2^X}$$

$$E[X] = \frac{E[G]}{(n-2)m_2^X - (n-3)}$$

$$\alpha = \left((n-1)m_2^X - (n-2)\right)\left((n-2)m_2^X - (n-3)\right)^2$$

$$\beta = (n-2)(m_2^X - 1)\left(n(n-1)(m_2^X)^2 - n(2n-5)m_2^X + (n-1)(n-3)\right)$$

The parameters μ_{x1}, μ_{x2} and p_x are calculated by solution a) with p=1 using the mean and the normalized moments of X calculated above.

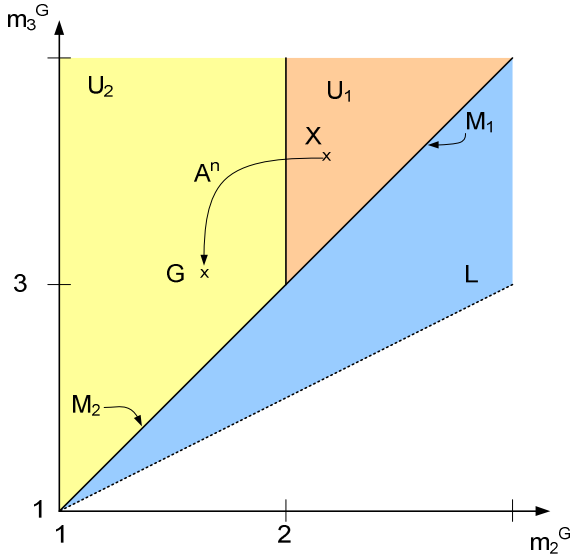


Figure 5-45: G is well-represented by $A^n(X)$ where X is a 2-phase Coxian distribution.

c) $G \in L$

G is well-represented by Z

$$Z = \begin{cases} W & with\ probability\ p \\ 0 & with\ probability\ 1-p \end{cases}$$

where W is an EC distribution with mean and normalized moments as follows:

$$p = \frac{1}{2m_2^G - m_3^G}$$

$$m_2^W = pm_2^G$$

$$m_3^W = pm_3^G$$

$$E[W] = \frac{E[G]}{p}$$

Observe that

$W \in M_1 \cup M_2$

If $W \in M_1$ case a) can be used for the solution using the normalized moments from the equations above. If $W \in M_2$ case b) can be used.

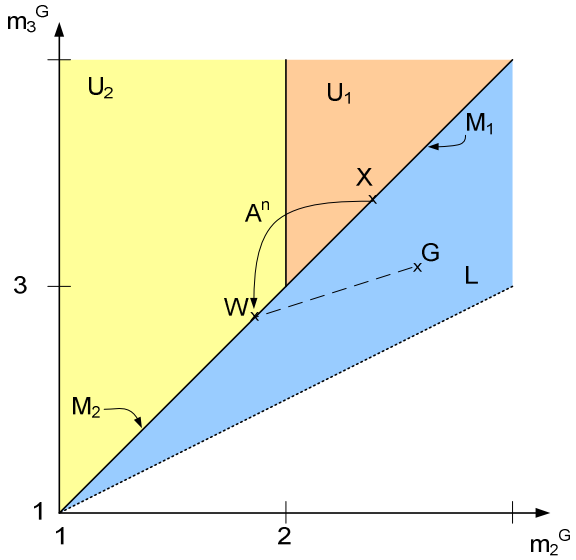


Figure 5-46: G is well-represented by Z, where Z is W=A^n(X) with probability p and 0 with probability 1-p and X is a 2-phase Coxian distribution

The number of phases of the EC distribution provided by the simple solution is at most OPT(G) + 2 (for a proof see [Osogami]).

5.5.1.2 Improved Closed-Form Solution

The improved closed-form solution of the OHB method provides a solution for all distributions G that can be well-represented by a PH distribution, in particular those where $G \notin \Phi$.

d) $G \in U \cap (\Phi)^C$

First find distribution W that is an EC distribution with the mean and the normalized moments as follows:

$W : W \in \Phi$

$$n = \frac{2m_2^G - 1}{m_2^G - 1},$$

$$m_2^W = \frac{1}{2}\left(\frac{n-1}{n-2} + \frac{n}{n-1}\right),$$

$$m_3^W = \frac{m_3^G}{m_2^G} m_2^W,$$

$$p_W = \frac{m_2^W}{m_2^G},$$

$$E[W] = \frac{E[G]}{p_W}$$

G is well-represented by Z for

$$Z = \begin{cases} W & with\ probability\ p_W \\ 0 & with\ probability\ 1 - p_W \end{cases}$$

The parameters of the solution for W are obtained by the simple closed-form solution.

e) $G \in M \cup L$

In this case the simple solution provides the parameters (n, p, μ_{x1}, μ_{x2}, p_x) except that

the number of phases n is calculated by equation (4). If n > 2, then n is decremented by one.

The improved closed-form solution is not numerically stable for

$G \in U \quad and$

$m_2^G \; is \; close \; to \; \frac{n+1}{n}$

Please refer to [Osogami] for a detailed analysis of numerical stability and methods to increase numerical stability by increasing the number of phases.

5.6 Source Model for a Terminal-Based Location Enabler

The OHB method presented in the presious section shall now be used to develop a source model for a terminal-based location enabler. The behaviour of mobile nodes that move at random was analyzed by a simulation presented earlier in this chapter (see “Mobility Simulation for a Terminal-Based Location Enabler”). The results of these simulations are the statistical properties of interarrival times of location update notifications that will arrive at a presence location server. It has been shown that the simulated distribution of interarrival times does not correspond to a simple negative-exponential distribution. The histogram of the simulated interarrival times and the chi-square analysis show, that the tail of the observed distribution is longer than with a negative exponential distribution.

The distribution of the simulatied data shows a heavy tail behaviour that is well known in telecommunications and computer science. In heavy tail distributions a high-frequency or high-amplitude population is followed by a low-frequency or low-amplitude population which gradually "tails off". This heavy tail characteristic can also be observed in real world data. The following histograms (see Figure 5-47 - Figure 5-50) show the interarrival times of location requests for Location Based Services of mobilkom austria with two different priorites.

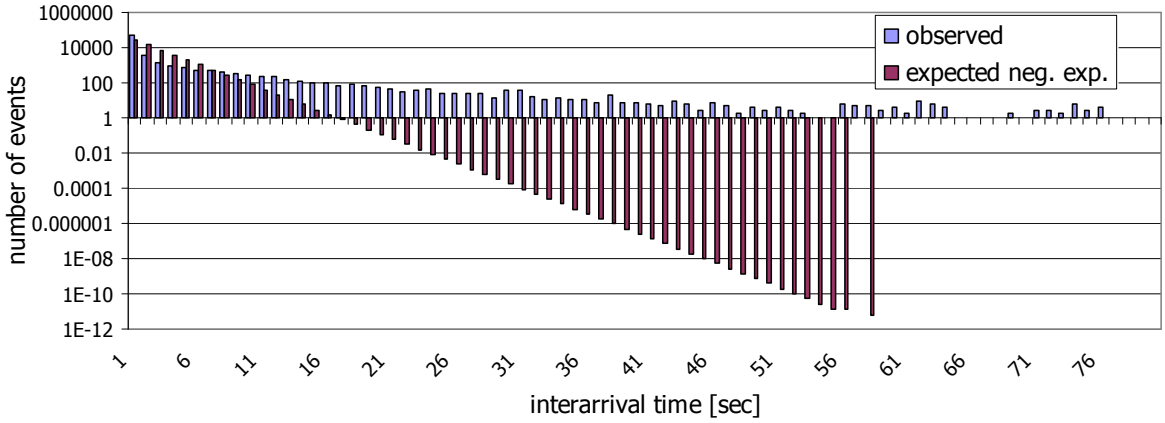


Figure 5-47: interarrival time histogram of a normal priority location service compared to a negative exponential distribution (logarithmic scale)

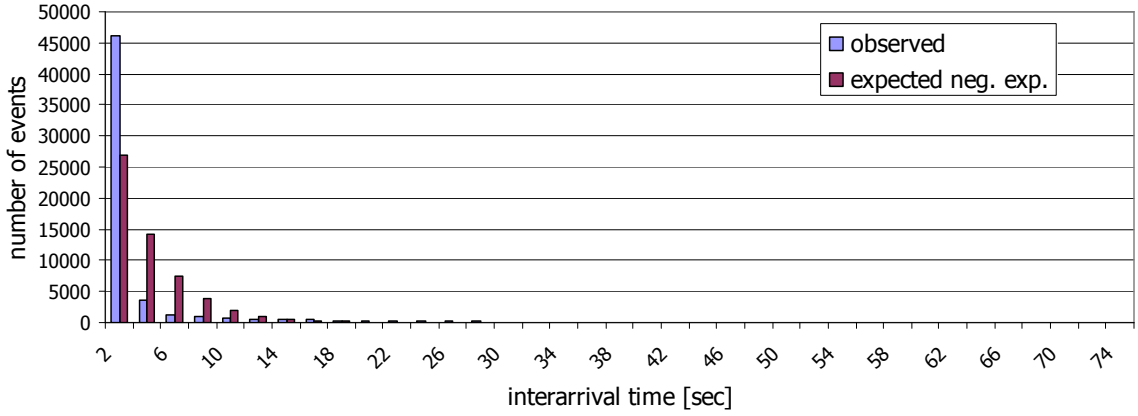


Figure 5-48: interarrival time histogram of a normal priority location service compared to a negative exponential distribution (linear scale)

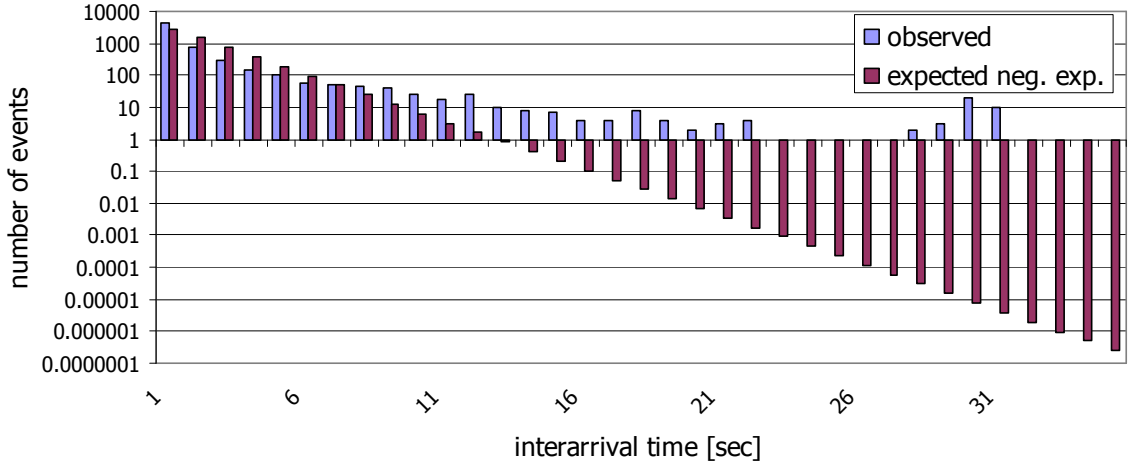


Figure 5-49: interarrival time histogram of a high priority location service compared to a negative exponential distribution (logarithmic scale)

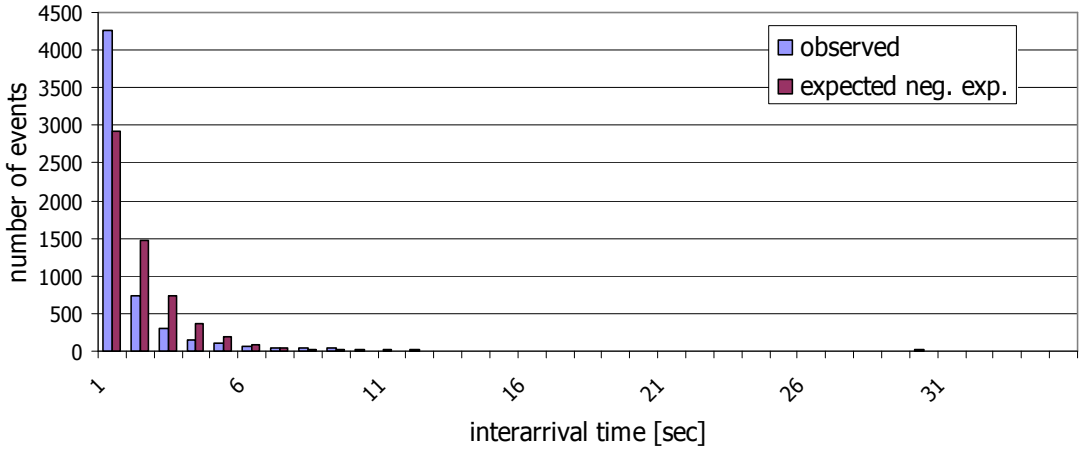


Figure 5-50: interarrival time histogram of a high priority location service compared to a negative exponential distribution (linear scale)

Table 5-9 shows the 1st, 2nd, and 3rd moments of the measured interarrival times of the location service requests with normal and high priority.

priority	1st moment	2nd moment	3rd moment
high	29.3329	4956.90	1883010.09
normal	3.0737	89.67	6546.39

Table 5-9: moments of the measured interarrival times of the location service requests with normal and high priority

The OHB-method is therefore chosen to approximate a phase-type model for general distributed interarrival times. The OHB-method results in a model matching the first three moments of the simulated interarrival times. The resulting model consists of negative-exponentially distributed phases and can thus be easily integrated into a server model. The general model is shown in Figure 5-51.

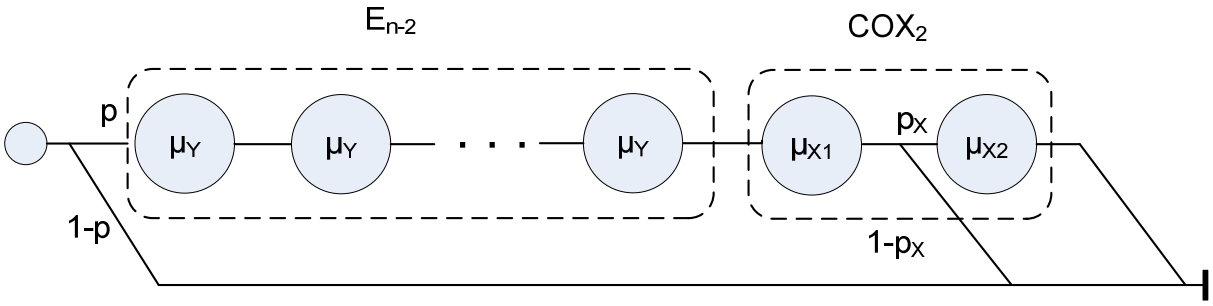


Figure 5-51: general EC distribution model of the OHB method

By appliying the OHB fitting algorithm to the moments of the measured interarrival times of the location service requests with normal and high priority (see Table 5-9), the parameters of a fitting EC distribution have been calculated in Table 5-10.

priority	1st normed moment	2nd normed moment	3rd normed moment	n	p	μ_y	p_x	μ_{x1}	μ_{x2}
high	29.33	5.76	12.95	2	1	0.007	0.113	0.071	0.007
normal	3.07	9.49	23.75	2	1	0.038	0.062	0.670	0.039

Table 5-10: parameters of an EC distribution for the measured interarrival times of the location service requests with normal and high priority

The calculated EC distribution parameters show, that 2 COX phases are sufficient to fit the measured data (n = 2) and that no traffic is branched when entering the model (p = 1). The source model for the measured data is shown in Figure 5-52.

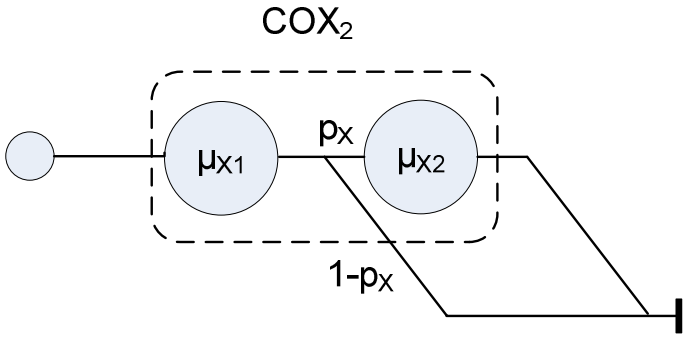


Figure 5-52: source model for for the measured interarrival times of the location service requests with normal and high priority

5.6.1 Combined Priority queue system model

The source characteristic of location requests that is modelled by a 2 phase Cox model in this section and the location presence server model with non preemptive priotity queues (see also Section 5.3.2) shall now be combined to a full system model.

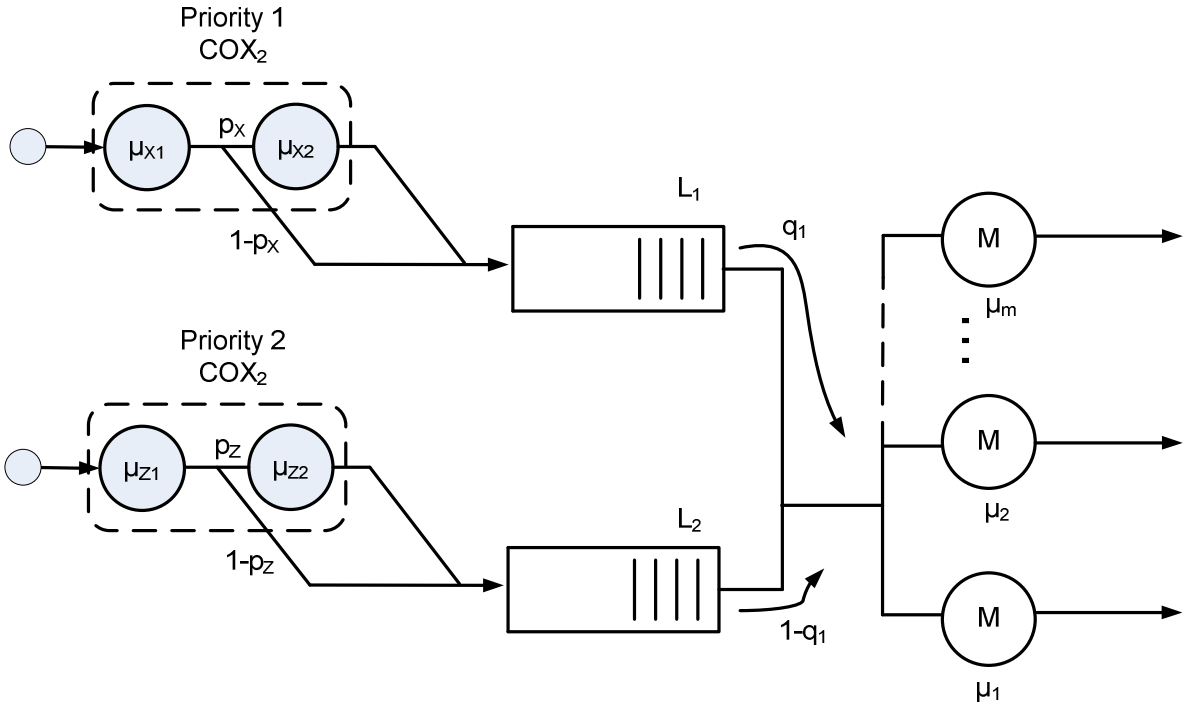


Figure 5-53: system model of a server with 2 non-preemptive priority queues and 2 phase Cox sources

Figure 5-53 shows a system with 2 priority queues. Jobs with priority i arrive at the system with a rate of λ_i. The system shall be analysed for a different number of servers m. Each server shall have the same service rate μ.
The traffic intensity u is defined as:

$$u = \frac{\sum_{i=1}^{n} \lambda_i}{m\,\mu}$$

The traffic rates intensity shall be varied in the simulation from 0.1 to 1. The rates are chosen in the following way:

$$m\mu = 1$$

$$\lambda_1 + \lambda_2 = um\mu = u$$

$$\lambda_1 = \frac{u}{10}$$

$$\lambda_2 = u - \lambda_1 = \frac{9}{10}u$$

This means that 10% of the traffic has priority 1.
The service rates of Cox phase are chosen approximated to the measured values from the previous section:

$$\mu_{X1} = 2\lambda_1$$

$$\mu_{X2} = 0.2\lambda_1$$

$$\mu_{Z1} = 2\lambda_2$$

$$\mu_{Z2} = 0.2\lambda_2$$

$$p_X = p_Z = 0.1$$

Note also, that by keeping mμ = 1, the total throughput of a multi threading system is simulated. The maximum server throughput is always kept constant at normalized value of 1, while this traffic throughput is divided among the simulated server processes. Thus the influence of having multiple server threads in parallel is analysed.
When all servers are busy, jobs with priority i are stored in a queue with L_i places. A job that arrives at a full queue is rejected and the system is called blocked for that priority. The queue lengths in this example are set to:

$L_1 = 5$

$L_2 = 10$

A server that has finished the current job takes a new job from one of the queues, if this queue is not empty. The highest priority i is chosen with a probability of q_i, if lower queues are not empty. In this example these probabilities are:

$q_1 = 0.8$

$q_2 = 1.0$

This means that if both queues are full a job is taken from queue 1 with probability 0.8.
The performance evaluation of the system was done using a software package developed at the Institute of Broadband Communication, Vienna University of Technology, that implements iterative numerical methods for the solution of Continuous Time Markov Chain systems [Sommereder].
The following tables (Table 5-11 - Table 5-13) and figures (Figure 5-54 - Figure 5-56) show the performance evaluation results for a location server system with two priotity queues and an input characterized by a Cox distribution.

u	E[X]	E[Q1]	E[Q2]	P[B1]	P[B2]	P[W2]	P[W2]
0.1	0.1196	0.0012	0.0184	1.530E-10	1.075E-09	0.1000	0.1000
0.2	0.2951	0.0049	0.0901	1.201E-08	1.647E-06	0.2000	0.2000
0.3	0.5656	0.0118	0.2539	1.520E-07	8.681E-05	0.3000	0.2999
0.4	0.9881	0.0219	0.5669	8.982E-07	1.050E-03	0.3993	0.3982
0.5	1.5991	0.0353	1.0685	3.466E-06	5.397E-03	0.4954	0.4900
0.6	2.3615	0.0515	1.7262	1.014E-05	1.611E-02	0.5837	0.5676
0.7	3.1841	0.0699	2.4538	2.440E-05	3.396E-02	0.6604	0.6264
0.8	3.9871	0.0899	3.1728	5.095E-05	5.740E-02	0.7243	0.6669
0.9	4.7279	0.1112	3.8402	9.558E-05	8.420E-02	0.7764	0.6923
1	5.3926	0.1335	4.4404	1.651E-04	1.125E-01	0.8185	0.7062

Table 5-11: Performance measures of a priority queue system with 2 priorities and 1 server and Cox input

u	E[X]	E[Q1]	E[Q2]	P[B1]	P[B2]	P[W1]	P[W2]
0.1	0.2056	0.0003	0.0053	4.3597E-11	3.1107E-10	0.0276	0.0276
0.2	0.4463	0.0024	0.0440	5.8592E-09	8.1228E-07	0.0942	0.0942
0.3	0.7683	0.0073	0.1610	9.6500E-08	5.5565E-05	0.1843	0.1842
0.4	1.2350	0.0160	0.4201	6.6578E-07	7.8335E-04	0.2877	0.2869
0.5	1.8943	0.0285	0.8733	2.8322E-06	4.4316E-03	0.3955	0.3910
0.6	2.7185	0.0444	1.5026	8.8189E-06	1.4063E-02	0.4984	0.4844
0.7	3.6159	0.0629	2.2250	2.2112E-05	3.0855E-02	0.5898	0.5589
0.8	4.4984	0.0833	2.9563	4.7438E-05	5.3563E-02	0.6668	0.6133
0.9	5.3157	0.1051	3.6453	9.0680E-05	8.0015E-02	0.7300	0.6501
1	6.0495	0.1280	4.2706	1.5875E-04	1.0828E-01	0.7813	0.6731

Table 5-12: Performance measures of a priority queue system with 2 priorities and 2 servers and Cox input

u	E[X]	E[Q1]	E[Q2]	P[B1]	P[B2]	P[W1]	P[W2]
0.1	0.3018	0.0001	0.0017	1.4232E-11	9.9705E-11	0.0087	0.0087
0.2	0.6245	0.0013	0.0232	3.1678E-09	4.3351E-07	0.0495	0.0495
0.3	1.0132	0.0050	0.1083	6.6074E-08	3.7658E-05	0.1231	0.1230
0.4	1.5355	0.0125	0.3243	5.2136E-07	6.0821E-04	0.2206	0.2200
0.5	2.2489	0.0241	0.7343	2.4079E-06	3.7409E-03	0.3305	0.3268
0.6	3.1359	0.0397	1.3342	7.8978E-06	1.2520E-02	0.4404	0.4278
0.7	4.1059	0.0582	2.0472	2.0479E-05	2.8440E-02	0.5403	0.5119
0.8	5.0642	0.0788	2.7853	4.4923E-05	5.0528E-02	0.6260	0.5756
0.9	5.9540	0.1009	3.4905	8.7182E-05	7.6688E-02	0.6971	0.6205
1	6.7527	0.1242	4.1359	1.5424E-04	1.0494E-01	0.7549	0.6501

Table 5-13: Performance measures of a priority queue system with 2 priorities and 3 servers and Cox input

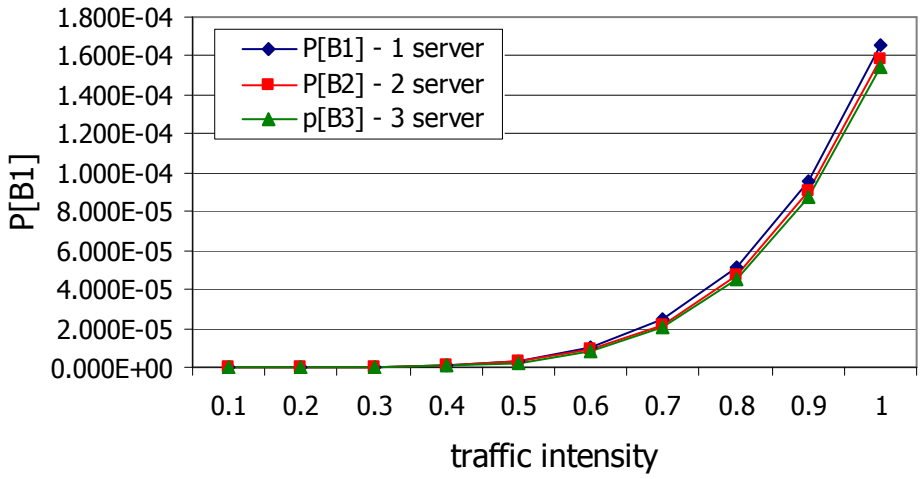


Figure 5-54: Blocking probability of priority 1 queue for different server scenarios and Cox input

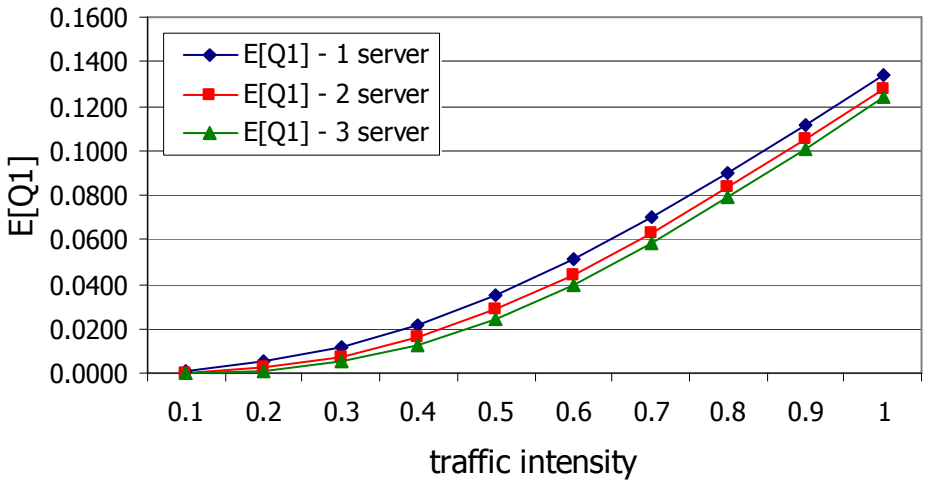


Figure 5-55: Excpected number of jobs in queue of priority 1 queue for different server scenarios and Cox input

Figure 5-56: Waiting probability of priority 1 queue for different server scen and Cox input

The performance results for a priority system with Cox input shall now be compared to the results in Section 5.3.3 , where the same system with negative exponentially distributed input was analyzed. The following figures (Figure 5-57 - Figure 5-59) show a comparison of the priority 1 traffic blocking probability P[B1], the priority 1 traffic waiting probability P[W1] and the priority 1 expeted queue length E[Q1] for a three server system.

The most outstanding difference between a negative exponentially distributed input traffic compared to a Cox distributed input traffic is that the blocking probability increases much faster with the traffic intensity. The expected queue length does not change much for the two input traffic distributions. The waiting probability for Cox distributed input traffic is higher for low traffic intensities and lower for high traffic intensities compared to negative exponentially distributed traffic.

As a concluding result it can be stated, that the distribution of the input traffic has significant influence on the performance measures of the studied non-preemptive priotity queueing system.

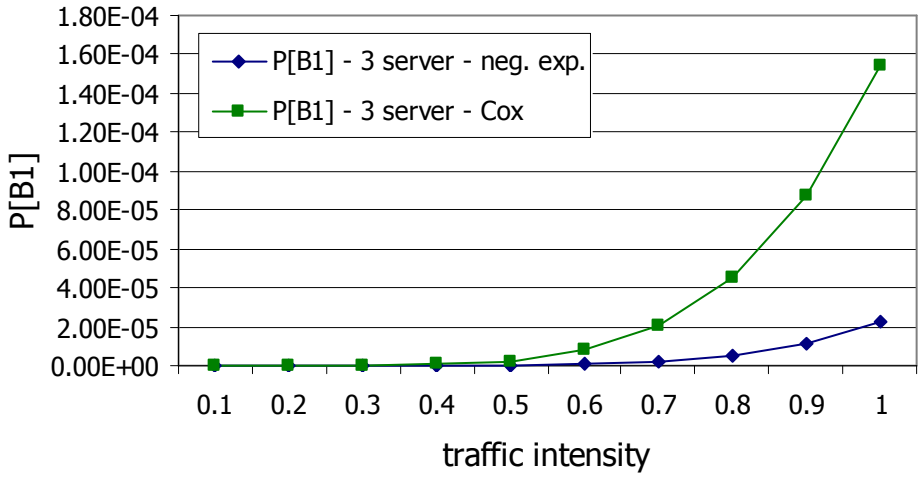


Figure 5-57: comparison of the blocking probability for priority 1 traffic for a 3 server system

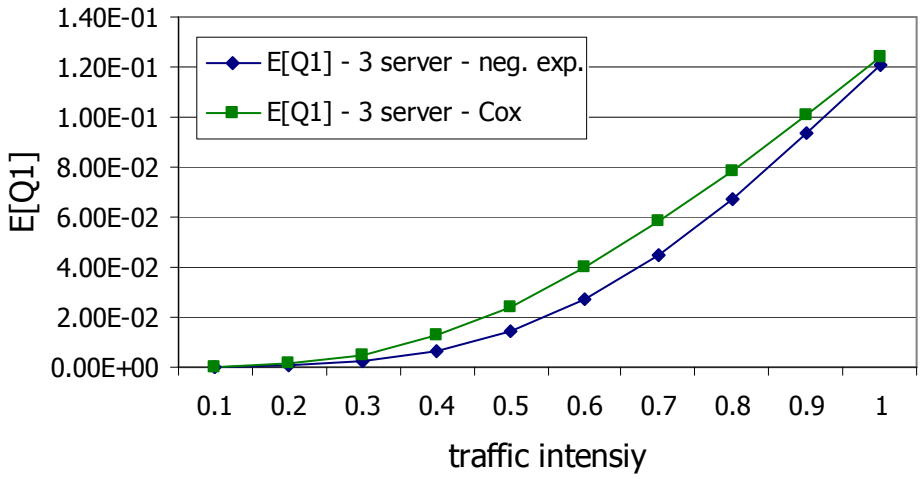


Figure 5-58: comparison of the expected queue length for priority 1 traffic for a 3 server system

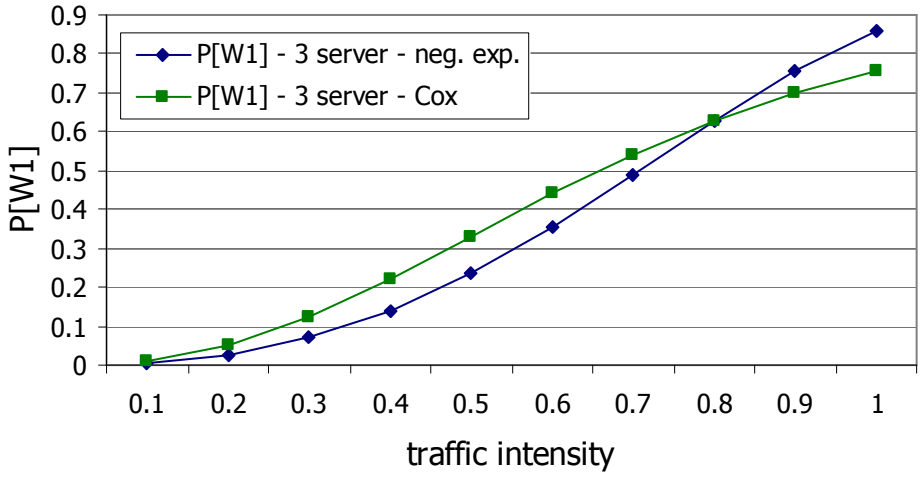


Figure 5-59: comparison of the waiting probability for priority 1 traffic for a 3 server system

6 Conclusion

This thesis covered the general aspects of the architecture and design of Next generation Service Platforms and discussed in detail specific service design issues in the development of value-added services on top of such platforms.

In the first part of this thesis, a service platform architecture for next generation networks was developed that blends the characteristics and requirements of classical telecommunication networks with the concepts and design principles of service oriented computing. The introduction of a service platform for the implementation and operation of multimedia services in next generation networks was proposed and the new technologies are presented and evaluated with the goal to specify a carrier-grade service framework for multimedia communications in heterogeneous networks. The proposed service platform architecture complies with Service Oriented Architecture (SOA) principles and a mapping of Session Initiation Protocol (SIP) functionality to Parlay services was proposed. Furthermore the separation of service logic from network technology was proposed by using the Parlay APIs to build a communication system as a "softswitch" application running on a service platform. A prototype of the proposed architecture has been implemented using the SIP Servlet API and was described in detail.

In the second part of this thesis, the validity of the architectural approach was proven, by showing how value added services in general and location based services in particular can be mapped to the proposed design patterns. Location Based Services (LBS) are chosen as a showcase in order to discuss architecture and design of value added services in service platforms. A patent of the author was presented, that shows how the proposed architecture can be used in combination with cryptographical methods to ensure the privacy of service platform users. This novel privacy mechanism is targeted for location and presence services in the 3G service architecture and uses cryptographic techniques well suited to run in small devices with little computing and power resources.

Furthermore a terminal-based location enabler based on the 3GPP IP Multimedia Subsystem (IMS) presence architecture was proposed. This novel location architecture for the IMS was based on SIP application layer signaling which supports both terminal generated and network-calculated location information. It is the main advantage of the proposed terminal-based location enabler that it supports triggered location updates in a simple and scalable way, so that necessary location updates are kept to an absolute minimum and scarce resources in the radio access network are used in the most economic way.

Finally the performance of the proposed location service architecture was analysed by simulation and by the analytical methods of queueing theory. A detailed performance analysis for a carrier-grade deployment of a Location Service (LCS) was conducted, based on the proposed presence location architecture. The gains in network capacity were presented and the performance values of the proposed architectiure were calculated.

7 Acronyms

3GPP	3rd Generation Partnership Project
AAA	Authentication, Authorisation and Accounting
ADSL	Asymmetric Digital Subscriber Line
API	Application Program(ming) Interface
APN	Access Point Name
APN-NI	APN Network Identifier
CAMEL	Customised Application For Mobile Network Enhanced Logic
CCM	CORBA Component Model
CERN	Centre Européen de Recherche Nucléaire
CGI	Common Gateway Interface
COPS	Common Open Policy Service
CS	Circuit Switched
CSCF	Call/Session Control Function
DCOM	Distributed Component Object Model
DNS	Domain Name System
EDGE	Enhanced Data for GSM Evolution
FDD	Frequency Division Duplex
FIPS	Federal Information Processing Standards
FTW	Forschungszentrum Telekommunikation Wien (Telecommunications Research Center Vienna)
GERAN	GSM EDGE Radio Access Network
GGSN	Gateway GPRS Support Node
GIOP	Generic Inter ORB Protocol
GMLC	Gateway Mobile Location Center
GPRS	General Packet Radio Service
GPS	Global Positioning System
GSM	Global System for Mobile Communications
HLR	Home Location Register
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
IDL	Interface Description Language (CORBA)
IETF	Internet Engineering Task Force
IIOP	Internet Inter-ORB Protocol (CORBA)
IMS	IP Multimedia Subsystem
IMSI	International Mobile Subscriber Identity
IN	Intelligent Network
IP	Internet Protocol
IPDL	Idle Period Downlink
ISDN	Integrated Services Digital Network
ISIM	IMS Subscriber Identity Module
ISO	International Organization for Standardization
ITU	International Telecommunication Union

JAIN ..... Java API for Integrated Networks
JMS ..... Java Messaging Services
JPEG ..... Joint Photographic Experts Group
JSP ..... Java Server Pages
JVM ..... Java Virtual Machine
LAN ..... Local Area Network
LBS ..... Location Based Services
LCS ..... Location Services
LIF ..... Location Interoperability Forum
LMU ..... Location Measurement Unit
MAC ..... Message Authentication Code
MEGACO ..... Media Gateway Control
MMS ..... Multimedia Messaging Service
MS ..... Mobile Station
MSC ..... Mobile services Switching Centre
MSISDN ..... Mobile Station ISDN Number
NAI ..... Network Access Identifier
NIST ..... National Institute of Standards and Technology
NGN ..... Next Generation Network
OASIS ..... Organization for the Advancement of Structured Information Standards
OMA ..... Open Mobile Alliance
OMG ..... Object Management Group
ORB ..... Object Request Broker
OSA ..... Open Service Architecture
OTDOA ..... Observed Time Difference Of Arrival
OTP ..... One-Time-Password
PAN ..... Personal Area Network
PDP ..... Packet Data Protocol
PET ..... Privacy Enhancement Technologies
PKI ..... Public Key Infrastructure
PLMN ..... Public Land Mobile Network
PS ..... Packet Switched
QoS ..... Quality of Service
RADIUS ..... Remote Authentication Dial In User Service
RAN ..... Radio Access Network
RMI ..... Remote Method Invocation
RNC ..... Radio Network Controller
RPC ..... Remote Procedure Call
RTCP ..... RTP Control Protocol
RTP ..... Real Time Transport Protocol
SCF ..... Service Capability Feature
SGML ..... Standard Generalized Markup Language
SGSN ..... Serving GPRS Support Node
SIM ..... Subscriber Identity Module
SIP ..... Session Initiation Protocol
SLA ..... Service Level Agreement

SLEE	Service Logic Execution Environment
SMS	Short Message Service
SOAP	SOAP (is not an acronym any longer)
SOC	Service Oriented Architecture
SSL	Secure Socket Layer
TINA-C	Telecommunications Information Networking Architecture Consortium
TCP	Transmission Control Protocol
TDD	Time Division Duplex
UE	User Equipment
UDDI	Universal Description, Discovery and Integration
UICC	Universal Integrated Circuit Card
UID	User Identity
UMTS	Universal Mobile Telecommunications System
URL	Universal Resource Locator
URI	Universal Resource Identifier
UTRAN	Universal Terrestrial Radio Access Network
UUID	Universally Unique Identifier
VLR	Visitor Location Register
VPN	Virtual Private Network
XML	Extensible Markup Language
W3C	World Wide Web Consortium
WGS-84	World Geodetic System 1984
WLAN	Wireless Local Area Network
WS-I	Web Service Interoperability Organization
WS-S	Web Services Security
WWW	World Wide Web

8 Literature

[3GPP] 3rd Generation Partnership Project (3GPP), http://www.3gpp.org

[3GPP LBS Priv] 3rd Generation Partnership Project, Technical Report, "Enhanced support for User Privacy in location services (Release 5)", 3GPP TR 23.871 V5.0.0 (2002-07) http://www.3gpp.org/ftp/Specs/html-info/23871.htm

[3GPP TS 22.071] 3rd Generation Partnership Project, Technical Specification Group Services and System Aspects, "Location Services (LCS), Service description, Stage 1 (Release 4)", 3GPP TS 22.071 V7.1.0 (2005-01) http://www.3gpp.org/ftp/Specs/html-info/22071.htm

[3GPP TS 23.002] 3rd Generation Partnership Project, Technical Specification Group Services and System Aspects, "Network Architecture", (Release 7), TS 23.002 V7.1.0 (2006-03) http://www.3gpp.org/ftp/Specs/html-info/23002.htm

[3GPP TS 23.141] 3rd Generation Partnership Project, "Presence Service; Architecture and Functional Description", 3GPP TS 23.141, Version 6.7.0, 09/2004 http://www.3gpp.org/ftp/Specs/html-info/23141.htm

[3GPP TS 23.218] 3rd Generation Partnership Project, Technical Specification Group Core Network, "IP Multimedia (IM) session handling, IM call model, Stage 2 (Release 6)", 3GPP TS 23.218 V6.4.0 (2006-06) http://www.3gpp.org/ftp/Specs/html-info/23218.htm

[3GPP TS 23.228] 3rd Generation Partnership Project, Technical Specification Group Services and System Aspects, "IP Multimedia Subsystem (IMS), Stage 2, (Release 6)", 3GPP TS 23.228 V6.13.0 (2006-03) http://www.3gpp.org/ftp/Specs/html-info/23228.htm

[3GPP TS 23.271] 3rd Generation Partnership Project, Technical Specification Group Services and System Aspects, "Functional stage 2 description of LCS (Release 4)", 3GPP TS 23.271 V7.0.0 (2005-03) http://www.3gpp.org/ftp/Specs/html-info/23271.htm

[3GPP TS 29.328] 3rd Generation Partnership Project, Technical Specification Group Core Network and Terminals, "IP Multimedia (IM) Subsystem Sh interface, Signalling flows and message contents, (Release 6)", 3GPP TS 29.328 V6.9.0 (2006-03) http://www.3gpp.org/ftp/Specs/html-info/29328.htm

[3GPP TS 29.329] 3rd Generation Partnership Project, Technical Specification Group Core Network and Terminals, "Sh Interface based on the Diameter protocol, Protocol details, (Release 6)", 3GPP TS 29.329 V6.7.0 (2006-09) http://www.3gpp.org/ftp/Specs/html-info/29329.htm

[3GPP TS 31.103] Technical Specification Group Core Network and Terminals, "Characteristics of the IP Multimedia Services Identity Module (ISIM) application (Release 6)", 3GPP TS 31.103 V6.12.1 (2007-06) http://www.3gpp.org/ftp/Specs/html-info/31103.htm

[3GPP TS 32.200] 3rd Generation Partnership Project, Technical Specification Group Services and System Aspects, "Telecommunication management, Charging

management, Charging principles (Release 5)", 3GPP TS 32.200 V5.9.0 (2005-09)
http://www.3gpp.org/ftp/Specs/html-info/32200.htm

[3GPP TS 32.240] 3rd Generation Partnership Project, Technical Specification Group Services and System Aspects, Telecommunication management, Charging management, Charging architecture and principles, (Release 6), 3GPP TS 32.240 V6.4.0 (2006-09)
http://www.3gpp.org/ftp/Specs/html-info/32240.htm

[3GPP TS 33.978] 3rd Generation Partnership Project, Technical Specification Group Services and System Aspects, "Security aspects of early IP Multimedia Subsystem (IMS), (Release 6)", 3GPP TR 33.978 V6.6.0 (2006-12)
http://www.3gpp.org/ftp/Specs/html-info/33978.htm

[Allen] Arnold O. Allen, "Probability, Statistics and Queueing Theory with Computer Science Applications", Academics Press Inc., 1978

[Axis] The Apache Web Services Project,
http://ws.apache.org/axis/

[Apache] The Apache Software Foundation,
http://www.apache.org/

[Belch] Gunter Belch, Stefan Greiner, Hermann de Meer, and Kishor S. Trivedi, "Queueing Networks and Markov Chains, Modeling and Performance Evaluation with Computer Science Applications", Wiley, 1998

[Bellare] Mihir Bellare, Ran Canetti, Hugo Krawczyk, "Message Authentication using Hash Functions – The HMAC Construction", RSA Laboratories' CryptoBytes, Vol. 2, No. 1, Spring 1996,
http://www.cs.ucsd.edu/users/mihir/papers/hmac-cb.pdf

[Bequet2001] Henry Bequet, "Professional Java SOAP", Wrox Press Ltd., 2001

[Bessler] S. Bessler, A.V. Nisanyan, K. Peterbauer, R. Pailer, J. Stadler, "A Service Platform for Internet-Telecom Services using SIP", Sixth International Conference on Intelligence in Networks (SMARTNET 2000), September 18-22, 2000, Vienna, Austria

[BonnMotion] A mobility scenario generation and analysis tool, University of Bonn,
http://web.informatik.uni-bonn.de/IV/Mitarbeiter/dewaal/BonnMotion/

[BPEL4WS] "Business Process Execution Language for Web Services Version 1.1", IBM developer Works, May 2003
http://www-106.ibm.com/developerworks/library/ws-bpel/

[Broch] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, J. Jetcheva, "A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols," Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking, Dallas, TX, October 1998

[Camarillo2006] G. Camarillo, M. A. Garcia-Martin, "The 3G IP Multimedia Subsystem (IMS), Second Edition", John Wiley & Sons Ltd, 2006

[CCM] "CORBA Components, Version 3, June 2002, formal/02-06-65", Object Management Group (OMG),
http://www.omg.org/cgi-bin/doc?formal/02-06-65

[Cline] A. K. Cline, C. B. Moler, G. W. Stewart, J. H. Wilkinson, "An Estimate for the Condition Number of a Matrix", SIAM Journal on Numerical Analysis, Vol. 16, No. 2 (Apr., 1979) , pp. 368-375

[Cuevas] A. Cuevas, J. I. Moreno, P. Vidales, H. Einsiedler, “The IMS Service Platform: A Solution for Next-Generation Network Operators to Be More than Bit Pipes”, IEEE Communications Magazine, August 2006, Vol. 44, No. 8

[Desrochers] Stephane Desrochers, Roch H. Glitho, Kindy Sylla, "Experimenting with Parlay in a SIP Environment: Early Results", IP Telecom Services Workshop 2000 (IPTS 2000) Atlanta, September 2000

[Dobbertin] Hans Dobbertin, “Cryptoanalysis of MD5 Compress”, May 2, 1996 http://citeseer.ist.psu.edu/68442.html

[Eckel TIP] Bruce Eckel, “Thinking in Patterns with Java“,Mind View Inc., May 2003 http://www.bruceeckel.com/Books/TIPatterns/

[Erl] T. Erl, “Service-Oriented Architecture: Concepts, Technology, and Design”, Prentice Hall, August 2005 Excerpt: http://www.looselycoupled.com/opinion/2005/erl-core-infr0815.html

[EU Data Priv] “Directive 2002/58/EC of the European Parliament and of the Council concerning the processing of personal data and the protection of privacy in the electronic communications sector (Directive on privacy and electronic communications)”, 12 July 2002, Official Journal of the European Communities L 201/37

[Fabini] J. Fabini, M. Happenhofer, R. Pailer, “Terminal-Centric Location Services for the IP Multimedia Subsystem”, IEEE 63rd Vehicular Technology Conference, May 2006, Melbourne, Australia

[FTW] Forschungszentrum Telekommunikation Wien, (Telecommunications Research Center Vienna), http://www.ftw.at

[FTW A1] Research project documentation, “Service Platforms and Interoperability”, Telecommunications Research Center Vienna (ftw.). http://www.ftw.at/ftw/research/projects/ProjekteFolder/A1

[FTW P2] Research project documentation, “Service Platforms”, Telecommunications Research Center Vienna (ftw.). http://www.ftw.at/ftw/research/projects/ProjekteFolder/P2

[Gartner] G. Gartner, W. Cartwright, M. P. Peterson (Eds.), “Lecture Notes in Geoinformation and Cartography, Location Based Services an TeleCartography”, Springer, 2007

[Geopriv] Charter of the Geographical Location and Privacy (GEOPRIV) working group of the Internet Engineering Task Force (IETF), http://www.ietf.org/html.charters/geopriv-charter.html

[Geopriv Req] J. Cuellar et al., “Geopriv Requirements”, IETF RFC 3693, http://www.ietf.org/rfc/rfc3693.txt

[Gbaguidi] C. Gbaguidi j. P. Hubaux g. Pacifici A.N. Tantawi , “Integration of Internet and telecommunications: An Architecture for Hybrid Services”, IEEE JSAC vol.17, no.9, Sept. 1999

[Gitman] Mitch Gitman, “Web Services for Interoperable Management”, FTPOnline, Fawcette Technical Publications, March 15 2004 http://www.ftponline.com/special/opsmgmt/gitman/default.asp

[GML] OpenGIS, "Open Geography Markup Language (GML) Implementation Specification", OGC 02-023r4, January 2003, http://www.opengis.org/techno/implementation.htm

[H.323] Telecommunication Standardization Sector of the International Telecommunication Union: Packet-Based Multimedia Communications Systems. Recommendation H.323. November 2000.

[Haller] N. Haller, C. Metz, P. Nesser, M. Straw, IETF RFC 2289, "A One-Time Password System", February 1998 http://www.ietf.org/rfc/rfc2289.txt?number=2289

[Hauser 01] C. Hauser, M. Kabatnik, "Towards Privacy Support in a Global Location Service", Proceedings of the IFIP Workshop on IP and ATM Traffic Management (WATM/EUNICE 2001), Paris, 2001 http://www.ind.uni-stuttgart.de/Content/Publications/Archive/Ha_EUNICE01_33986.pdf

[HTTP 1.1] R. Fielding, J. Gettys , J. Mogul ,H. Frystyk, L. Masinter, P. Leach,T. Berners-Lee, IEFT RFC 2616 "Hypertext Transfer Protocol -- HTTP/1.1", Internet Engineering Task Force, June 1999, http://www.ietf.org/rfc/rfc2616.txt

[IDL2Java] OMG Specification, "IDL to Java Language Mapping Specification", Version 1.2, formal/02-08-05, August 2002 http://www.omg.org/docs/formal/02-08-05.pdf

[IMS] 3rd Generation Partnership Project, Technical Specification Group Services and System Aspects, "IP Multimedia Subsystem (IMS)", Stage 2, Release 6, TS 23.228 V6.13.0 (2006-03) http://www.3gpp.org/ftp/Specs/html-info/23228.htm

[IN] ITU-T, "Q12xx, Intelligent Network", ITU-T Standards

[IP] J. Postel (Ed.), Internet Protocol. IETF RFC 791. September 1981 http://www.ietf.org/rfc/rfc791.txt

[IPDC] I. Elliott, "IPDC: Media Control Protocol", draft-elliott-ipdc-media-00.txt, IETF Internet Draft, August 1998. http://tools.ietf.org/html/draft-elliott-ipdc-media-00

[IPSEC] S. Kent, R. Atkinson, "Security Architecture for the Internet Protocol", IETF RFC 2401, November 1998 http://www.ietf.org/rfc/rfc2401.txt

[ISDN] Telecommunication Standardization Sector of the International Telecommunication Union: Integrated Services Digital Networks (ISDNs). Recommendation I.120. March 1993.

[ISO 15408] International Organization for Standardization, "Common Criteria for Information Technology Security Evaluation, Part 2: Security functional requirements", January 2004, Version 2.2, CCIMB-2004-01-002, aligned with ISO 15408, http://www.commoncriteria.de/it-security_english/ccinfo.htm

[ITU-T NGN] ITU, International Telecommunication Union, Study Group 13, Next generation Networks http://www.itu.int/ITU-T/studygroups/com13/index.asp

[JAIN] Sun Microsystems, JAIN APIs, JSR-22, http://java.sun.com/products/jain

[JAIN SIP] Sun Microsystems, JAIN SIP API Specification, JSR-32 http://jcp.org/en/jsr/detail?id=32

[Java] Sun Microsystems, Java programming langague, http://java.sun.com

[JavaEE] Sun Microsystems, Java Enterprise Edition, http://java.sun.com/javaee/5/

[J2ME] Sun Microsystems, Java 2 Mobile Edition - J2ME Personal Profile, JSR-000062 Personal Profile Specification ,Version 1.0, http://java.sun.com/products/personalprofile/index.jsp

[JAXRPC] Java Community Process JSR-101, "Java API for XML based RPC", Sun Microsystems, October 2003, http://java.sun.com/xml/jaxrpc/index.jsp

[Johnson] C. Johnson, H. Joshi, J. Khalab, "WCDMA radio network planning for location services and system capacity", 3G Mobile Communication Technologies, 2002. Third International Conference on (Conf. Publ. No. 489), May 2002, pp. 340- 344

[Jorns] O. Jorns, S. Bessler, R. Pailer, "An efficient mechanism to ensure location privacy in telecom service applications", IFIP TC6 Conference for Network Control and Engineering for QoS, Security and Mobility (NetCon 2004), 11-2004, Mallorca, Spain

[JSR-179] Java Specification Request JSR-179, "Location API for the Java 2 Platform, Micro Edition (J2ME) version 1.0", Nokia, 2003. http://jcp.org/en/jsr/detail?id=179

[KERBEROS] J. Kohl and C. Neuman, "The Kerberos Network Authentication Service (V5)," RFC 1510, September 1993, http://www.ietf.org/rfc/rfc1510.txt

[Kleinrock] Lenoard Kleinrock, "Queueing Systems, Volume I: Theory", Wiley, 1975

[Kreyszig] E. Kreyszig, "Advanced Engineering Mathematics", 6th edition, Wiley, 1988

[Küpper] A. Küpper, G. Treu, "From Location to Position Management: User Tracking for Location-based Services", Kommunikation in Verteilten Systemen (KiVS) Kurzbeiträge und Workshop 2005, pp. 81-88

[Lamport] Leslie Lamport, "Password Authentication with Insecure Communication", Communications of the ACM, vol. 24(11), 1981, p. 770-772. http://research.microsoft.com/users/lamport/pubs/password.pdf

[LIF] Location Inter-operability Forum (LIF), "Mobile Location Protocol Specification", Version 3.0.0, June 2002 http://www.openmobilealliance.org/tech/affiliates/lif/lifindex.html

[Mahy] R. Mahy, "Document Format for Filtering and Reporting Location Notications in the Presence Information Document Format Location Object (PIDF-LO)", Proceedings of the Sixty-Sixth Internet Engineering Task Force, Montreal, Canada, July 9-14, 2006 http://www3.ietf.org/proceedings/06jul/IDs/draft-ietf-geopriv-loc-filters-00.txt

[Mampaey 00a] M. Mampaey, A. Couturier, "Using TINA Concepts for IN Evolution", IEEE Communications, June 2000

[Mampaey 00b] M. Mampaey, “TINA for Services and Advanced Signaling and Control in Next-Generation Networks”, IEEE Communications, October 2000

[Manione 99a] R. Manione, P. Renditore, “A TINA Light Service Architecture for the Internet-Telecom Scenario”, TINA 99 Conference, Hawaii, USA, April 1999

[Manione 99b] R. Manione, M. Festa, P. Renditore, D. Sereno, M. Spaziani, “Service Issues in Web Call Centers”, Telecom 99 Forum, Geneva, 1999

[MEGACO] F. Cuervo, N. Greene, A. Rayhan, C. Huitema, B. Rosen, J. Segers: “Megaco Protocol Version 1.0”, IETF RFC 3015, November 2000

[Menasce 2002] Daniel A. Menasce, Virgilio A. F. Almeida, “Capacity Planning for Web Services”, Prentice Hall, 2002

[MD5] Ronald L. Rivest, “The MD5 Message-Digest Algorithm”, IETF RFC 1321, April 1992, http://www.ietf.org/rfc/rfc1321.txt?number=1321

[MIME] N. Freed, N. Borenstein, “Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies”, IETF RFC 2045, Internet Engineering Task Force, November 1996. http://www.ietf.org/rfc/rfc2045.txt

[MGCP] M. Arango, A. Dugan, I. Elliott, C. Huitema, S. Pickett, “Media Gateway Control Protocol (MGCP) Version 1.0”, IETF RFC 2705, October 1999 http://www.ietf.org/rfc/rfc2705.txt

[Mosmondor] Mosmondor, M.; Skorin-Kapov, L.; Kovacic, M., “Bringing location based services to IP multimedia subsystem”, Electrotechnical Conference, 2006, MELECON 2006, IEEE Mediterranean 16-19 May 2006,Page(s):746 – 749, Digital Object Identifier 10.1109/ MELCON.2006.1653207

[OASIS] OASIS consortium, Organization for the Advancement of Structured Information Standards http://www.oasis-open.org

[OMA] Open Mobile Alliance http://www.openmobilealliance.org

[OMA AP] Open Mobile Alliance (OMA), “OMA Architecture Principles V1.2”, OMA-ArchitecturePrinciples-V1_2-20040414-A, Approved Version 1.2 – 14 Apr 2004

[OMA MLP] Open Mobile Alliance (OMA), “Mobile Location Protocol (MLP)”, Candidate Version 3.1 – 16 Mar 2004

[OMA OSPE] Open Mobile Alliance (OMA), “OMA Service Provider Environment Architecture”, OMA-AD-OSPE-V1_0-20060411-D, Draft Version 1.0 – 11APR 2006

[OMA PEEM] Open Mobile Alliance (OMA), “Policy Evaluation, Enforcement and Management Architecture”, OMA-AD-Policy_Evaluation_Enforcement_Management-V1_0-20060312-D, Draft Version 1.0 – 12 Mar 2006

[Open ORB] The Community Open ORB Project, http://openorb.sourceforge.net/

[OSA] 3GPP, "Open Services Architecture", Application Programming Interface”, 3GPP TS 29.198, TR 29.998,

http://www.3gpp.org/ftp/Specs/html-info/29198.htm
http://www.3gpp.org/ftp/Specs/html-info/29998.htm

[Osogami] T. Osogami and M. Harchol-Balter, “A closed-form solution for mapping general distributions to minimal PH distributions”, Technical Report CMU-CS-03-114, School of Computer Science, Carnegie Mellon University, 2003

[Pailer 01] Pailer R., Stadler J., “A service framework for carrier grade multimedia services using Parlay APIs over a SIP System”, ACM Wireless Mobile Internet Workshop WMI2001, Rome, Italy, July 2001

[Pailer 03] R. Pailer, J. Stadler, I. Miladinovic, “Using PARLAY APIs over a SIP System in a Distributed Service Platform for Carrier Grade Multimedia Services”, ACM Wireless Network journal, Volume 9 (2003) No. 4

[Pailer Pat 04] DI Rudolf Pailer, Mag. Oliver Jorns, DI Dr. Sandford Bessler, „Verfahren zur Umwandlung von Target-Ortsinformationen in Mehrwertinformationen“, Patentanmeldung beim Österreichischen Patentamt, A 363/2004

[Pailer 05] R. Pailer, F. Wegscheider, J. Fabini, “Terminal-Centric Location Services in the IP Multimedia Subsystem”, 3rd Symposium on LBS & TeleCartography, Vienna University of Technology, Vienna, Austria, November 2005

[Pailer 06] R. Pailer, S. Bessler, F. Wegscheider, “A Terminal-Based Location Service Enabler for the IP Multimedia Subsystem”, IEEE Wireless Communications and Networking Conference April 2006 (WCNC06), Las Vegas, USA

[Ratti] C. Ratti, A. Sevtsuk, S. Huang, R. Pailer, “Mobile Landscapes: Graz in Real Time”, 3rd Symposium on LBS & TeleCartography, Vienna University of Technology, Vienna, Austria, November 2005

[Reichl 06a] P. Reichl, S. Bessler, J. Fabini, R. Pailer, J. Zeiss, “Implementing a Native IMS Location Service Enabler over a Prototypical IMS Core Network Testbed”, The Third IEEE International Workshop on Mobile Commerce and Wireless Services (WMCS'06) June 26, 2006, San Francisco, USA

[Reichl 06b] P. Reichl, S. Bessler, J. Fabini, R. Pailer, A. Poropatich, N. Jordan, R. Huber, H. Weisgrab, C. Brandner, I. Gojmerac, M. Ries, F. Wegscheider, “Practical Experiences with an IMS-Aware Location Service Enabler on Top of an Experimental Open Source IMS Core Implementation”, Journal of Mobile Multimedia, Rinton Press, Fall 2006

[PAP2003] M.P. Papazoglou and D. Georgakopoulos, “Service Oriented Computing”, Communications of the ACM, October 2003/Vol. 46, No. 10

[Parlay] The Parlay Group, http://www.parlay.org/

[Parlay Mob] The Parlay Group, “Open Service Access (OSA);Application Programming Interface (API);Part 6: Mobility SCF (Parlay 4)”, ETSI ES 202 915-6 V1.2.1, Aug 2003

[ParlayX] The Parlay Group, “Parlay 4.0, Parlay X Web Services Specification”, Verion 1.0, May 2003
http://www.parlay.org/specs/Parlay_X_Web_Services_Specification_V1_0.zip

[ParalyX Loc] The Parlay Group, “Parlay X Web Services; Part 9: Terminal Location”, 3GPP TS 29.199-09, Version 6.0.0, 09/2004, http://www.3gpp.org/ftp/Specs/archive/29_series/29.199-09/29199-09-600.zip

[ParlayX Pres] The Parlay Group, “Parlay X Web Services; Part 14: Presence”, 3GPP TS 29.199-14, Version 6.0.0, 09/2004, http://www.3gpp.org/ftp/Specs/archive/29_series/29.199-14/29199-14-600.zip

[Patterns] E. Gamma, R. Helm, R. Johnson, J. Vlissides, “Design Patterns, Elements of Reusable Object-Oriented SoftwareAddison-Wesley, January 1995

[Pavlovski] C. J. Pavlovski, “Service Delivery Platforms in Practice”, IEEE Communications Magazine, March 2007, Vol. 45, Nr. 3

[Peterbauer] K. Peterbauer, J. Stadler, I. Miladinovic, T. Pudil, "SIP Servlet API Extensions", IETF Draft, draf-peterb-sip-servlet-ext-00.txt, http://www.ikn.tuwien.ac.at/ftw-a1/draft-peterbauer-sip-servlet-ext-00.txt

[Polk] J. M. Polk, B. Rosen, “Requirements for Session Initiation Protocol Location Conveyance”, IETF Internet-Draft, draft-ietf-sipping-location-requirements-02.txt, October 25th, 2004

[RAPID IST] RAPID - Work Package 2 - Stream 6: PETs in Infrastructure, FP5 IST Roadmap project, RAPID – IST-2001-38310, “Roadmap for Advanced Research in Privacy and Identity Management” http://www.ra-pid.org/

[RFC 2026] S. Bradner, “The Internet Standards Process -- Revision 3”, Best Current Practice IETF RFC 2026, October 1996

[RFC 2401] S. Kent, R. Atkinson, „Security Architecture for the Internet Protocol“, IETF Standards Track RFC 2401, November 1998

[RFC 2486] B. Aboba, M. Beadles, “The Network Access Identifier”, IETF Standards Track RFC 2486, January 1999

[RFC 2748] D. Durham (Ed.), J. Boyle, R. Cohen, S. Herzog, R. Rajan, A. Sastry, “The COPS (Common Open Policy Service) Protocol”, IETF Standards Track RFC 2748, January 2000

[RFC 2778] M. Day, J. Rosenberg, H. Sugano, “A Model for Presence and Instant Messaging”, IETF RFC 2778, February 2000

[RFC 2806] A. Vaha-Sipila, “URLs for Telephone Calls”, IETF Standards Track RFC 2806, April 2000.

[RFC 2821] J. Klensin (Ed.), „Simple Mail Transfer Protocol“, IETF Stadards Track RFC 2821, April 2001

[RFC 2865] C. Rigney, S. Willens, A. Rubens, W. Simpson, “Remote Authentication Dial In User Service (RADIUS)”, IETF Standards Track RFC 2865, June 2000

[RFC 2976] S. Donovan, “The SIP INFO Method”, IETF Standards Track RFC 2976, October 2000.

[RFC 3015] F. Cuervo, N. Greene, A. Rayhan, C. Huitema, B. Rosen, Marconi, J. Segers, “Megaco Protocol Version 1.0”, IETF Standards Track RFC 3015, November 2000

[RFC 3050] J. Lennox, H. Schulzrinne, J. Rosenberg, „Common Gateway Interface for SIP“, IETF Informational RFC 3050, January 2001

[RFC 3262] J. Rosenberg and H. Schulzrinne, “Reliability of Provisional Responses in the Session Initiation Protocol (SIP)”, IETF Standards Track RFC 3262, June 2002.

[RFC 3263] J. Rosenberg, H. Schulzrinne, „Session Initiation Protocol (SIP): Locating SIP Servers“, IETF Standards Track RFC 3263, June 2002

[RFC 3265] A. B. Roach, “Session Initiation Protocol (SIP)-Specific Event Notification”, IETF Standards Track RFC 3265, June 2002

[RFC 3323] J. Peterson, “A Privacy Mechanism for the Session Initiation Protocol (SIP)”, IETF Standards Track RFC 3323, November 2002

[RFC 3428] B. Campbell, J. Rosenberg, H. Schulzrinne, C. Huitema, and D. Gurle, “Session, Initiation Protocol (SIP) Extension for InstantMessaging“, IETF Standards Track RFC 3428, December 2002.

[RFC 3455] M. Garcia-Martin, E. Henrikson, D. Mills, “Private Header (P-Header) Extensions to the Session Initiation Protocol (SIP) for the 3rd-Generation Partnership Project (3GPP)”, IETF Informational RFC 3455, January 2003

[RFC 3588] P. Calhoun, J. Loughney, E. Guttman, G. Zorn, J. Arkko, “Diameter Base Protocol”, IETF Standards Track RFC 3588, September 2003

[RFC 3725] J. Rosenberg, J. Peterson, H. Schulzrinne, G. Camarillo, „Best Current Practices for Third Party Call Control (3pcc) in the Session Initiation Protocol (SIP)“, IETF Best Current Practice RFC 3725, April 2004

[RFC 3840] J. Rosenberg, H. Schulzrinne, P. Kyzivat, „Indicating User Agent Capabilities in the Session Initiation Protocol (SIP)“, IETF Standards Track RFC 3840, August 2004

[RFC 3841] J. Rosenberg, H. Schulzrinne, P. Kyzivat, „Caller Preferences for the Session Initiation Protocol (SIP)“, IETF Standards Track RFC 3841, August 2004

[RFC 3856] J. Rosenberg, ‘A Presence Event Package for the Session Initiation Protocol (SIP)’, IETF Standards Track RFC 3856, August 2004

[RFC 3857] J. Rosenberg, “A Presence Event Package for the Session Initiation Protocol (SIP)”, IETF Standards Track RFC 3857, August 2004

[RFC 3858] J. Rosenberg, “An Extensible Markup Language (XML) Based Format for Watcher Information”, IETF Standards Track RFC 3858, August 2004

[RFC 3863] H. Sugano, S. Fujimoto, G. Klyne, A. Bateman, W. Carr, J. Peterson, “Presence Information Data Format (PIDF)”, IETF Standards Track RFC 3863, August 2004.

[RFC 3880] J. Lennox, X. Wu, H. Schulzrinne, „Call Processing Language (CPL): A Language for User Control of Internet Telephony Services“, IETF Standards Track RFC 3880, October 2004

[RFC 4079] J. Peterson, “A Presence Architecture for the Distribution of GEOPRIV Location Objects”, IETF Informational RFC 4079, July 2005

[RFC 4419] ”, J. Peterson, “A Presence-based GEOPRIV Location Object Format”, IETF Stadards Track RFC 4119, December 2005

[RFC 4480] H. Schulzrinne, V. Gurbani, P. Kyzivat, J. Rosenberg, "RPID: Rich Presence: Extensions to the Presence Information Data Format (PIDF)", IETF Standards Track RFC 4480, July 2006

[RFC 4660] H. Khartabil, E. Leppanen, M. Lonnfors, J. Costa-Requena, "Functional Description of Event Notification Filtering", IETF Standards Track RFC 4660, September 2006

[RFC 4661] Khartabil, H., Leppanen, E., Lonnfors, M., and J. Costa-Requena, "An Extensible Markup Language (XML)-Based Format for Event Notification Filtering", IETF Standards Track RFC 4661, September 2006.

[RFC 4662] A. Roach, B. Campbell, J. Rosenberg, "A Session Initiation Protocol (SIP) Event Notification Extension for Resource Lists", IETF Standards Track RFC 4662, August 2006

[RTP] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, „RTP: A Transport Protocol for Real-Time Applications“, IETF Standards Track RFC 3550, July 2003

[SDP] Handley, V. Jacobson, "SDP: Session Description Protocol", IETF Standard Track RFC 2327, April 1998

[Servlet] Sun Microsystems, JSR-000154, Java Servlet 2.4 Specification, Nov 2003
http://java.sun.com/products/servlet/reference/api/index.html

[SGCP] M. Arango, C. Huitema, “Simple Gateway Control Protocol (SGCP) Version 1.1”, draft-huitema-sgcp-v1-1.txt, Draft. IETF Internet Draft. July 1998
http://www.argreenhouse.com/SGCP/sgcp-v1-1.html

[SHA-1] National Institute of Standards and Technology, “Secure Hash Standard”, Federal Information Processing Standards (FIPS) Publication 180-2, August 2002
http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf

[Shaham] R. Shaham, H. Schulzrinne, W. Kellerer, S.Thakolsri, “An Architecture for Location-based Service Mobility Using the SIP Event Model”, International Conference on Mobile Systems Application and Services, (Mobys 2004) Boston 2004

[SIMPLE] IETF Working Group “SIP for Instant Messaging and Presence Leveraging Extensions (simple)”,
http://www.ietf.org/html.charters/simple-charter.html

[SIP] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler, “SIP: Session Initiation Protocol”, IETF RFC 3261, June 2002,
http://www.rfc-editor.org/rfc/rfc3261.txt

[SIP Servlet] Sun Micorsystems, Java Specification Request JSR 116, “SIP Servlet API”,
http://www.jcp.org/en/jsr/detail?id=116

[SOAP 1.2] W3C Recommendation, “SOAP Version 1.2 Part 0: Primer”, “SOAP Version 1.2 Part 1: Messaging Framework”, “SOAP Version 1.2 Part 2: Adjuncts”, World Wide Web Consortium, June 2003,
http://www.w3.org/TR/2003/REC-soap12-part0-20030624/

http://www.w3.org/TR/2003/REC-soap12-part1-20030624/
http://www.w3.org/TR/2003/REC-soap12-part2-20030624/

[Sommereder] M. Sommereder, "Modellierung von Warteschlangensystemen mit dynamisch erzeugten Markov-Ketten, Diplomarbeit am Institut für Breitbandkommunikation der Technischen Universität Wien", Februar 2007

[Stadler] J. Stadler, I. Miladinovic, R. Pailer, M. Vucic, "Comparision of Softswitch-Architectures using Parlay as Application Interface", IFIP 7th Conference on Intelligence in Networks (SmartNet 2002), April 2002, Saariselkä, Lapland, Finland

[TCP] J. Postel (Ed.), "Transmission Control Protocol", IETF RFC 793. September 1981
http://www.rfc-editor.org/rfc/rfc793.txt

[TINA-C] Telecommunications Information Networking Architecture Consortium – TINA-C, "Service Architecture", Version 5.0, June 1997
http://www.tinac.com/

[TIPHON] Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON), "Network Architecture and Reference Configurations; Phase II: Scenario 1 + Scenario 2", TS 101 313 V0.4.2, February 1999.

[TISPAN NGN] ETSI Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN), "NGN Release 1; Release definition NGN R1", Doc. Nb. TR 180 001 Ver. 1.1.1 Ref. DTR/TISPAN-00001-NGN-R1, 2006-03-30

[UDDI] Uddi.org, OASIS Consortium, "UDDI Technical White Paper", September 2000,
http://www.uddi.org/whitepapers.html

[UDDI API] Uddi.org, OASIS Consortium, "UDDI Version 2.04 API Specification", UDDI Committee Specification, 19 July 2002
http://uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.pdf

[UDDI Data] Uddi.org, OASIS Consortium, "UDDI Version 2.03 Data Structure Reference", UDDI Committee Specification, 19 July 2002
http://uddi.org/pubs/DataStructure-V2.03-Published-20020719.pdf

[UDP] J. Postel (Ed.), "User Datagram Protocol", IETF RFC 768, August 1980
http://www.rfc-editor.org/rfc/rfc768.txt

[Vinoski 04] Steve Vinoski, "Web Services Notifications", IEEE Internet Computing, Volume 8, Number 2, March/April 2004,
http://www.iona.com/hyplan/vinoski/pdfs/IEEE-Web_Services_Notifications.pdf

[Vogels 03] Werner Vogels, "Web Services Are Not Distributed Objects", IEEE Internet Computing, Volume 7, Number 6, November/December 2003
http://weblogs.cs.cornell.edu/AllThingsDistributed/archives/000120.html

[Waldo94] Jim Waldo, Samuel C. Kendall, Ann Wollrath, Geoff Wyant, "A Note on Distributed Computing", Sun Microsystems Laboratories, November 1994,
http://research.sun.com/techrep/1994/abstract-29.html

[Weera 02] Sanjiva Weerawarana, Francisco (Paco) Curbera, "Business Process with BPEL4WS: Understanding BPEL4WS, Part 1", IBM developerWorks, 1 August 2002,

http://www-106.ibm.com/developerworks/webservices/library/ws-bpelcol1/

[WGS-84] World Geodetic System 1984 (WGS 84), "WGS 84 Implementation Manual", Version 2.4, EUROCONTROL, European Organization for the Safety of Air Navigation Brussels, Belgium, February 1998

[Wingfoot] Wingfoot Mobile Web Services Platform
http://www.wingfoot.com/index.jsp

[Winterbottom] J. Winterbottom, M. Thomson, H. Tschofenig, "GEOPRIV PIDF-LO Usage Clarification, Considerations and Recommendations", IETF Internet Draft, draft-ietf-geopriv-pdif-lo-profile-05.txt, October 23, 2006
http://www.ietf.org/internet-drafts/draft-ietf-geopriv-pdif-lo-profile-05.txt

[WS-Adr] A. Bosworth, et al., "Web Services Addressing (WS-Addressing)", joint specification by BEA Systems, IBM, and Microsoft, Mar. 2004;
ftp://www6.software.ibm.com/software/developer/library/ws-add200403.pdf

[WS Arch] David Booth et al., "Web Services Architecture", W3C Working Group Note, 11 February 2004,
http://www.w3c.org/TR/ws-arch/

[WSDL 2.0] W3C Working Draft, "Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language", March 2004, "Web Services Description Language (WSDL) Version 2.0 Part 2: Message Exchange Patterns", March 2004, "Web Services Description Language (WSDL) Version 1.2 Part 3: Bindings", June 2003, World Wide Web Consortium,
http://www.w3.org/TR/wsdl20/
http://www.w3.org/TR/wsdl20-patterns/
http://www.w3.org/TR/wsdl12-bindings/

[WSDM] OASIS, "Web Services Distributed Management: Management of Web Services (WSDM-MOWS 0.5)", OASIS Committee Draft, 2 April 2004
http://www.oasis-open.org/committees/download.php/6255/cd-wsdm-mows-0.5-20040402.pdf

[WS-R] OASIS, "Web Services Reliable Messaging TC WS-Reliability", OASIS Working Draft 0.995, 29 April 2004
http://www.oasis-open.org/committees/download.php/6569/WS-Reliability-2004-04-29.pdf

[WS-S] OASIS, "Web Services Security: SOAP Message Security 1.0 (WS-Security 2004)", OASIS Standard 200401, March 2004
http://www.oasis-open.org/committees/download.php/6367/oasis-200401-wss-soap-message-security-1.0.pdf

[X509] S. Santesson, et al, "Internet X.509 Public Key Infrastructure Qualified Certificates Profile,"
http://www.itu.int/rec/recommendation.asp?type=items&lang=e&parent=T-REC-X.509-200003-I

[XCAP] J. Rosenberg, "The Extensible Markup Language (XML) Configuration Access Protocol (XCAP)", draft-ietf-simple-xcap-12, IETF Internet-Draft, October 13, 2006
http://www.ietf.org/internet-drafts/draft-ietf-simple-xcap-12.txt

[XCAP DIFF] J. Rosenberg, "An Extensible Markup Language (XML) Document Format for Indicating Changes in XML Configuration Access Protocol (XCAP) Resources", IETF Internet-Draft, raft-ietf-simple-xcap-diff-04, October 17, 2006
http://www.ietf.org/internet-drafts/draft-ietf-simple-xcap-diff-04.txt

[XCAP LIST] J. Rosenberg, "Extensible Markup Language (XML) Formats for Representing Resource Lists", IETF Internet-Draft, draft-ietf-simple-xcap-list-usage-05, February 7, 2005
http://www.ietf.org/internet-drafts/draft-ietf-simple-xcap-list-usage-05.txt

[XML 1.0] World Wide Web Consortium, "Extensible Markup Language (XML) 1.0 (Third Edition)", W3C Recommendation, February 2004,
http://www.w3.org/TR/2004/REC-xml-20040204/

[XMLENC] World Wide Web Consortium, "XML Encryption Syntax and Processing", W3C Working Draft, 04 March 2002,
http://www.w3.org/TR/xmlenc-core/

[XMLDEC] World Wide Web Consortium, "Decryption Transform for XML Signature", W3C Recommendation, 10 December 2002,
http://www.w3.org/TR/xmlenc-decrypt

[XML Infoset] World Wide Web Consortium, "XML Information Set", W3C Recommendation, 24 October 2001,
http://www.w3.org/TR/xml-infoset/

[XML NS] World Wide Web Consortium, "Namespaces in XML 1.1", W3C Recommendation, February 2004
http://www.w3.org/TR/xml-names11/

[XML Schema] World Wide Web Consortium, "XML Schema Part 0: Primer", "XML Schema Part 1: Structures", "XML Schema Part 2: Datatypes", W3C Recommendation, May 2001,
http://www.w3.org/TR/xmlschema-0/
http://www.w3.org/TR/xmlschema-1/
http://www.w3.org/TR/xmlschema-2/

[XMLSIG] World Wide Web Consortium, W3C Recommendation, "XML Signature Syntax and Processing," February 2002,
http://www.w3.org/TR/xmldsig-core/

[Yoon] J. Yoon, M. Liu, and B. Noble, "Random waypoint considered harmful", Proceedings of IEEE Infocom '03, pages 1312-1321, San Francisco, California, USA, April 2003

[Zimmermann] O. Zimmermann, P. Krogdahl, C. Gee, "Elements of Service-Oriented Analysis and Design", IBM developers works, 02 June 2004,
http://www-128.ibm.com/developerworks/webservices/library/ws-soad1/

[Zundt] Zundt, M.; Deo, G.; Naumann, M.; Ludwig, M. ",De-centralized location management: minimizing privacy concerns for location based services", Information Technology: Research and Education, 2005, ITRE 2005. 3rd International Conference on 27-30 June 2005 Page(s):23 – 27, Digital Object Identifier 10.1109/ITRE.2005.1503057

9 List of Figures

10 List of Tables

Appendix A: Location Presence Internet Draft

An IETF Internet-Draft specifying a *locpres* event package has been published by our research group. This *locpres* event package describes a location presence architecture for SIP that allows a watcher to subscribe to notifications about the geographical location of a presentity. The *locpres* event package is a concrete instantiation of the general event notification framework defined in [RFC 3265].

Internet-Drafts that has been remained unchanged in the IETF Internet-Draft directory for more than six months without being recommended by the Internet Engineering Steering Group (IESG) for publication as an "Request for Comments" (RFC), is simply removed from the Internet-Drafts directory [RFC 2026].

At the time of writing this thesis, it cannot be predicted whether the published Internet-Draft will develop into an RFC. Therefore the text of the locpres event package Internet-Draft is printed here as Appendix A.

SIMPLE
Internet-Draft
Intended status: Standards Track
Expires: September 21, 2007

R. Pailer
mobilkom austria AG

March 20, 2007

A Location Presence Event Package for the Session Initiation Protocol (SIP)

draft-pailer-locpres-00.txt

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on September 21, 2007.

Copyright Notice

Abstract

This document describes the usage of the Session Initiation Protocol (SIP) for subscriptions and notifications of location presence. Presence is defined as the willingness and ability of a user to communicate with other users on the network. Presence data is general information about the state of a user like "on-line" and "off-line". Location presence extends conventional presence by taking the physical location of a user into account. Location

presence is information about the geographical position of a user. Subscriptions and notifications of location presence are supported by defining an event package within the general SIP event notification framework. This protocol is also compliant with the Common Presence Profile (CPP) framework.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [1].

Table of Contents

1. Introduction

This document specifies a location presence event package for the Session Initiation Protocol (SIP) [2] according to the general SIP event notification framework [3]. Location presence data is information about the geographical position of a user. The location presence protocol builds on the concepts of the general SIP presence event package [4], but extends conventional presence by specifying entities and rules for the processing of subscriptions and notifications of location presence data.

Location presence data is needed in order to implement Location Based Services (LBS). LBS provide added value by taking into account the physical position of mobile users. Location presence data may consist of plain geographical coordinates, access point cell IDs, civil location in form of postal addresses or more abstract definitions like 'in the office', 'at home'. Examples of services are a map showing the user's current location or changing the handling of incoming calls when the user enters a specified area.

The system architecture of the location presence event package is based on the following two assumptions:

1. The optimal source for geographical location data is the user's terminal. Typically a GPS module is used to obtain location information. A network-based location method may be used as a fall-back option.

2. Geographical location is a special type of presence information that is referred to as 'location presence data'.

The fundamental assumption that leads to the reuse of the presence architecture is that location information is regarded as a kind of presence information [5]. The position of a user may in general be related to his presence state. The access to both presence and location data has the same requirements regarding security and privacy.

Location presence data however shows some essential differences from conventional presence data. Location data is semantically different from classic presence data. Presence attributes, defined as the Rich Presence Information Data Format (RPID) in RFC4480 [6] such as location type, activity, sphere, etc. can each take a small set of values, whereas geographical location has a continuous range of values, limited only by location data accuracy. This requires different handling of subscriptions for location presence data.

Whereas for conventional presence all watchers subscribe to changes that occur to a subset of attributes, for location data each watcher application may define different criteria for triggering events. Hence, publishing and storing location information on servers like normal presence information is not useful. Location data may also originate from different sources than normal presence state. The architecture of the location presence event package takes these differences into account.

This document defines a Location Presence Agent (LPA). The LPA is a Presence Agent for location data that is co-located with the Presence User Agent in the user's mobile terminal. It is the task of the LPA to process SUBSCRIBE requests for location presence data and to provide location information in NOTIFY messages. As the LPA is a software component running in the user's terminal, direct access to location information (e.g. from a GPS module) is possible. This means also that location data is distributed peer-to-peer between the watcher and the presentity and that the user has full control over the distribution of sensitive location data.

Notifications for conventional presence are sent out, when the presence status of the presentity changes. In the case of location data however, there is no predefined set of states. Therefore the watcher has to describe in the body of the SUBSCRIBE message the trigger conditions when the LPA should send out NOTIFY messages. Otherwise the LPA would have to send notifications independently of the actual need for that data. Trigger conditions for location data is described in Appendix A and includes triggers, if a presentity has moved a specified distance in horizontal or vertical direction, if a presentity has exceeded a specified velocity, or if a presentity enters/exits a certain area. Trigger conditions are necessary to reduce the cardinality of the location presence state space. Reducing the amount of notifications to a minimum is of utmost importance for mobile applications, because sending messages over a wireless link uses up battery and needs access network capacity. It is the most important advantage of a terminal-based location method that it supports triggered location notifications in a simple and scalable way. This means that the trigger logic has to be implemented in the terminal in form of an LPA.

The computing capacity of a mobile terminal will restrict the number and complexity of location presence data trigger criteria that can be processed in parallel. This document therefore specifies the possibility to identify a specific trigger condition by a URL in an 'xlink:href' attribute. A presentity may publish a list of predefined trigger conditions that a watcher can consequently subscribe to. A predefined trigger condition that is named and identifiable by a URI is a geographical tag defined by the

presentity. Unlike fixed presence states defined in RPID, trigger conditions can be created dynamically so that notifications for location presence data can be built around a user defined folksonomy of geotags.

In order to provide location presence data for terminals that are not equipped with a GPS module, or for terminals that are outside of GPS coverage, a Location Presence Data Aggregator (LPDA) component is specified in this document that aggregates location presence data from different sources. An alternative location data source may be a network based location enabler such as a 3GPP LCS [7]. Network based location enablers are specified for 2G and 3G wireless networks but offer in general a level of location data accuracy (~150 m) that is significant inferior to satellite based systems like GPS. Furthermore most deployed network based location enablers do not support triggered location updates so that polling schemes have to be used to track users which does not scale well. The LPDA acts as a state agent for the presentity and may issue subscriptions to several location data sources.

The location presence event package is compliant with the Common Presence Profile (CPP) framework [8]. It builds on the architecture defined for conventional presence like privacy mechanisms, security considerations, watcher information or authorization data. The location presence event package however considers location data specific requirements regarding subscription handling, location data sources, location data format and location data aggregation.

2. Definitions

This document uses the terms as defined in RFC2778 [9] and RFC3856 [4].
Additionally, the following terms are defined and/or additionally clarified:

Location Presence Agent (LPA): The LPA is a Presence Agent (PA) for location data that is co-located with the Presence User Agent (PUA) in the user's mobile terminal. A PA that is co-located with a PUA is also referred to as Edge Presence Server in [4]. It is the task of the LPA to process SUBSCRIBE requests for location presence data and to provide location information in NOTIFY messages.

Location Presence Data Aggregator (LPDA): The LPDA aggregates location presence data from different location data sources. The LPDA is a Presence Server acting as a state agent that is not co-located with the PUA. The LPDA may act as a subscriber and send

SUBSCRIBE requests to several location data sources, in particular towards a LPA.

3. Relation to SIP Presence RFC 3856

The location presence event package is a SIP event package according to RFC 3265 [3] that integrates location enabler with SIP presence specified in RFC 3856 [4]. A location enabler is any kind of positioning method that allows to query the geographical location of a user's terminal. This specification identifies data about the physical presence at a certain place to be a specialized form of general presence information and refers to it as 'location presence data'. Location presence data has the same requirements regarding authorization, authentication and security as conventional presence. It is the intention of this specification to extend RFC 3856 [4] rather than to define a separate presence system. In particular it is assumed that authorization of location presence subscriptions uses the same framework and infrastructure as conventional presence.

Location presence data is however sufficiently different from conventional presence in order to justify the specification in a location presence event package. There are several reasons for that.

First location services usually target mobile terminals. It is the goal of this specification to restrict location presence data formats to a complexity that can be handled by mobile terminals with restricted computing resources and limited battery capacity. This specification extends conventional presence by defining location presence data content formats.

Second it is of utmost importance to limit the use of wireless access capacity to the absolute minimum. A subscriber SHOULD avoid to poll a presentity for location presence data change. A notifier SHOULD avoid to flood the network with location presence data. The location presence data event package extends conventional presence by specifying a filter document format that allows a systematic control of notification rates.

Third notification rates can only be controlled effectively at the source of location presence data. Thus subscriptions including filter criteria have to be routed to the PA in control. In case of a LPA that is co-located with the PUA, the filters are executed in the mobile terminal. This means that the routing of location presence subscriptions will most probably be different from routing of subscriptions for conventional presence. By specifying an own event package name, subscriptions and notifications for location presence data can be routed independently of conventional presence.

4. Overview of Operation

A subscriber that wants to get notifications about the location presence of a presentity, sends a SUBSCRIBE request to the presentity. The Request-URI of the SUBSCRIBE requests identifies the presentity by its SIP URI, SIPS URI or presence (pres) URI. The SUBSCRIBE request may also carry a filter document in its body, that describes, when a location presence notification should be sent. The request is routed to a presence agent that is acting on behalf of the presentity. This specification defines two dedicated presence agents. The LPA is co-located with the PUA in order to accommodate positioning methods built into the terminal (e.g. a GPS module). LPDA is used to aggregate location data that originates from different positioning methods.

The presence agent first authenticates the subscription, then authorizes it. It is expected that the same authorization mechanism like for conventional presence is used. If authorized, a 200 OK response is returned. If authorization could not be obtained at this time, the subscription is considered "pending", and a 202 response is returned. In both cases, the PA sends an immediate NOTIFY message containing the location presence state of the presentity and of the subscription. Location presence data is transported in a PIDF-LO document [10], that may be empty in the case of a pending subscription or may contain some bogus information in order to protect the privacy of the presentity.

The presence agent will send NOTIFY messages when the user's location changes sufficiently to trigger a location presence data update. Changes in the state of the subscription itself can also trigger NOTIFY requests; that state is carried in the Subscription-State header field of the NOTIFY, and would typically indicate whether the subscription is active, pending or terminated.

The initial SUBSCRIBE message establishes a "dialog" with the presence agent. The subscription persists for a duration that is negotiated as part of the initial SUBSCRIBE. The subscriber will need to refresh the subscription before its expiration, if he wishes to retain the subscription. This is accomplished by sending a SUBSCRIBE refresh within the same dialog established by the initial SUBSCRIBE.

The subscriber can terminate the subscription by sending a SUBSCRIBE with an Expires header field value of zero. This causes an immediate termination of the subscription. A NOTIFY request is then generated by the presence agent with the current location. An initial SUBSCRIBE with an Expires value of zero can be used as a one-time location fetch.

The presence agent can terminate the subscription at any time by sending a NOTIFY request with a Subscription-State header field indicating that the subscription has been terminated.

5. Location Presence Event Package

This document specifies the Location Presence SIP event package according to RFC 3265 [3]. [3] defines a SIP extension framework for subscribing to, and receiving notifications of, events. It is the task of a specific event package to define the concrete events on top of this generic SIP event framework. This section defines the specific part of the Location Presence event package as required in [3].

5.1. Event Package Name

The name of this package is 'locpres'. As specified in [3], the package name is used in the Event header field of SUBSCRIBE and NOTIFY requests.

5.2. Event Package Parameters

The Location Presence event package does not define any additional parameters.

5.3. SUBSCRIBE Bodies

A SUBSCRIBE request in the location presence event package MAY contain a body. The request-URI, which identifies the presentity, combined with the event package name 'locpres' is sufficient for SIP request routing.

A subscription for location presence with an empty body is equivalent to a request for the current location of the presentity. The notifier MAY answer this request, depending on privacy policies, with a NOTIFY message, containing the current position of the presentity.

If a subscriber wants to control the conditions, when a notification is sent by the notifier, it MAY add a filter document to the SUBSCRIBE body. The default format for filter documents complying to this specification is 'application/ location-presence-delta-filter+xml' (see Appendix A) and all notifiers complying to this specification MUST be able to process such filters.

The notifier MAY provide a list of acceptable filter documents by means not specified in this document. A subscriber may for example

follow a URL given in an 'xlink:href' attribute to download a 'gml: extentOf' description of an area. If such a provided 'gml:Polygon' element contains a 'gml:id' attribute, this 'gml:id' attribute SHOULD be sent with the filter document. If the provided 'gml:Polygon' element contains a 'gml:name' element this 'gml:name' element SHOULD be sent with the filter document.

An initial SUBSCRIBE request establishes a SIP dialog with the PA. Subsequent SUBSCRIBE messages may be sent by the subscriber to refresh a subscription. These SUBSCRIBE refreshes MAY have an empty body without invalidating the filter document sent in the initial SUBSCRIBE. A subscriber MUST NOT change a filter specification in a subsequent SUBSCRIBE.

5.3.1. SUBSCRIBE Body examples

This section shows XML instance documents of typical SUBSCRIBE bodies.

The following XML instance document is an example of a subscription that specifies a movement filter. The presentity should only send out notification events, if the current location differs by 5 meters horizontally or vertically compared to the last notification.

```
<?xml version="1.0" encoding="UTF-8"?>
<p:location-filter xmlns:gml="http://www.opengis.net/gml"
  xmlns:p="urn:ietf:params:xml:ns:location-presence-filter"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <movedHoriz uom="urn:ogc:def:uom:EPSG::9001">5</movedHoriz>
  <movedVert uom="urn:ogc:def:uom:EPSG::9001">5</movedVert>
</p:location-filter>
```

The following XML instance document is an example of a subscription that specifies a containment filter. The presentity should only send out notification events on enter or exit of the specified area.

```
<?xml version="1.0" encoding="UTF-8"?>
<p:location-filter xmlns:gml="http://www.opengis.net/gml"
  xmlns:p="urn:ietf:params:xml:ns:location-presence-filter"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <enterOrExit>
    <gml:extentOf>
      <gml:Polygon gml:id="C648AF9A54"
        srsName="urn:ogc:def:crs:EPSG::4326">
        <gml:name>myOffice</gml:name>
        <gml:exterior>
          <gml:LinearRing>
            <gml:pos>48.222761 16.369345</gml:pos>
            <gml:pos>48.222518 16.371018</gml:pos>
            <gml:pos>48.221049 16.369951</gml:pos>
            <gml:pos>48.220977 16.369135</gml:pos>
            <gml:pos>48.222761 16.369345</gml:pos>
          </gml:LinearRing>
        </gml:exterior>
      </gml:Polygon>
    </gml:extentOf>
  </enterOrExit>
</p:location-filter>
```

The following XML instance document is an example of a subscription that specifies a containment filter. The presentity should only send out notification events on enter or exit of the area referenced by an xlink.

```
<?xml version="1.0" encoding="UTF-8"?>
<p:location-filter xmlns:gml="http://www.opengis.net/gml"
  xmlns:p="urn:ietf:params:xml:ns:location-presence-filter"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xlink="http://www.w3.org/1999/xlink">

  <enterOrExit>
    <gml:extentOf xlink:href="http://www.example.com/A9D82F6987" />
  </enterOrExit>
</p:location-filter>
```

5.4. Subscription Duration

Notifications for locations presence are controlled by location presence filter documents in the body of the SUBSCRIBE message. A NOTIFY message is sent, when a trigger condition of the filter is met, e.g. if a presentity has moved a specified distance in horizontal or vertical direction, if a presentity has exceeded a specified velocity, or if a presentity enters/exits a certain area.

According to [3] the subscriber MAY specify a subscription expiration time in the Expires header of the SUBSCRIBE message. If no Expires header is present in the SUBSCRIBE message, the following default values MUST be used:

- if the SUBSCRIBE message contains a valid location presence filter document, the default expiration is 3600 seconds.

- if the SUBSCRIBER message does not contain a valid location presence filter document, the default expiration time is 0 seconds.

5.5. NOTIFY Bodies

The body of a notification of the location presence event package contains a presence document. This document contains the geographical data describing the location presence of the presentity that the watcher subscribed to. All subscribers and notifiers MUST support the "application/location-presence-pidf+xml" presence data format as described in this document. The subscribe request MAY contain an Accept header field. If no such header field is present, it has a default value of "application/location-presence-pidf+xml". If the header field is present, it MUST include "application/ location-presence-pidf+xml", and MAY include any other types capable of representing location presence.

The content of presence information data containing location information is further specified as a PIDF Location Object (PIDF-LO) in [10]. PIDF-LO refers to the Geography Markup Language (GML) 3.0 [14] for coding location information and in particular to the GML 'feature.xsd' XML schema. GML is a thorough and versatile system for modeling all manner of geographic object types, topologies, metadata, coordinate reference systems and units of measurement. Since the publication of the PIDF-LO specification, the Open Geospatiale Consortium has published GML version 3.1 [11] that deprecates some elements used in PIDF-LO.

The target system for a location presence implementation is most likely a mobile terminal. In order to reduce the complexity of

implementing a notifier complying to this location presence event package specification, a notifier is REQUIRED to implement the restrictions and updates to the original PIDF-LO specification as follows:

- The implementation MUST use GML 3.1 [11] for PIDF-LO locations.
- Locations in GML MUST be encoded either as a 'gml:position' element for point geometries or a 'gml:extentOf' element for surface geometries. The 'gml:position' or 'gml:extentOf' elements MAY refer to the geometry elements by giving an URL in an 'xlink:href' attribute.
- The implementation MUST use either the GML 3.1 geometry elements 'gml:Point' or 'gml:Polygon' for encoding locations.
- The 'gml:Polygon' geometry MUST have not more than 4 unique points, therefore having not more than 5 points in total, building a general quadrangle. The points MUST be encoded as 'gml:pos' elements contained in a 'gml:LinearRing' element inside a 'gml:exterior' element.
- The location presence notifier MUST set the 'srsName' attribute of the 'gml:Point' or the 'gml:Polygon' geometry to 'urn:ogc:def:crs:EPSG::4326' [15]. The European Petroleum Survey Group (EPSG) geodetic parameter dataset [16] specifies coordinate reference system (CRS) definitions that describe geographical positions unambiguously. 'EPSG:4326' is a common geographic projection that refers to WGS84 (latitude, longitude) coordinates in degrees with Greenwich as the central meridian. Latitude is an angular measurement ranging from 0o at the Equator to +90o at the north pole and −90o at the south pole. Longitude is given as an angular measurement ranging from 0o at the central meridian to +180o eastward and −180o westward. Geographical coordinates MUST be encoded in 'gml:pos' elements using WGS84 notation (latitude followed by longitude) and the unit of measurement MUST be decimal degrees.
- The 'gml:Point' or the 'gml:Polygon' MAY contain a 'gml:id' attribute. If present, the 'gml:id' value MUST be unique for the presentity identified by the entity attribute of the 'presence' element.
- The 'gml:Point' or the 'gml:Polygon' MAY contain a 'gml:name' element.

The notifier MUST indicate presence at a location in one of the following formats:

- The position is given directly in a PIDF-LO 'location-info' element in form of a 'gml:position' or a 'gml:extendOf' element. The 'location-info' element MAY be empty in case the presentity does not want to reveal its position.

- The position is given in a PIDF-LO 'location-info' element, containing a 'pidfResource' element defined in Appendix A.2.1 that lists a set of 'containment' status elements . Each containment element MUST contain a 'gml:position' or a 'gml:extendOf' element as described above.

A notifier MUST use 'containment' status elements to encode location presence, if the subscriber provided an 'application/ location-presence-delta-filter+xml' filter document containing an 'enterOrExit' element.

A notifier that wants to aggregate location data that was acquired from different sources MUST add a separate 'tuple' element to the presence information document for each location data set. The PIDF-LO 'geopriv' element in each tuple MUST use a PIDF-LO 'method' element for each location and the positioning methods MUST NOT be the same for any tuple.

A location presence event package notifier MAY use location encoding schemes other than specified above, but the subscriber MUST request such a format explicitly in an Accept header.

A notifier that does not want to reveal its location to the subscriber MAY send an empty PIDF-LO 'location-info' element.

Further versions of this specification may refer to the documents [17] and [18] regarding restrictions and interoperability considerations for the use of GML inside PIDF-LO documents. [17] for example defines its own 'Circle' element (a center point and a radius) that is missing in the basic feature.xsd schema of GML.

5.5.1. NOTIFY body examples

This section shows XML instance documents of typical NOTIFY bodies.

The following XML instance document is an example of a notification that encodes a PIDF-LO location in a GML 3.1 point geometry. The GPS coordinates given in the 'pos' element are for Vienna, Austria.

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:presence entity="pres:presentity@example.com"
  xmlns:p="http://www.w3.org/XML/1998/namespace"
  xmlns:tns="urn:ietf:params:xml:ns:pidf"
  xmlns:gp="urn:ietf:params:xml:ns:pidf:geopriv10"
  xmlns:bp="urn:ietf:params:xml:ns:pidf:geopriv10:basicPolicy"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <tns:tuple id="03783453">
    <tns:status>
      <gp:geopriv>
        <gp:location-info>
          <gml:position>
            <gml:Point srsName="urn:ogc:def:crs:EPSG::4326">
              <gml:pos>48.208481 16.372601</gml:pos>
            </gml:Point>
          </gml:position>
        </gp:location-info>
         <gp:usage-rules>
          <bp:retransmission-allowed>false
          </bp:retransmission-allowed>
        </gp:usage-rules>
        <gp:method>GPS</gp:method>
      </gp:geopriv>
    </tns:status>
  </tns:tuple>
</tns:presence>
```

The following XML instance document is an example of a notification that encodes a PIDF-LO location in GML 3.1. The actual geometry element is linked by an 'xlink:href' attribute.

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:presence entity="pres:presentity@example.com"
  xmlns:p="http://www.w3.org/XML/1998/namespace"
  xmlns:tns="urn:ietf:params:xml:ns:pidf"
  xmlns:gp="urn:ietf:params:xml:ns:pidf:geopriv10"
  xmlns:bp="urn:ietf:params:xml:ns:pidf:geopriv10:basicPolicy"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <tns:tuple id="03783453">
    <tns:status>
      <gp:geopriv>
        <gp:location-info>
          <gml:extendOf
            xlink:href="http://www.example.com/A9D82F6987"/>
        </gp:location-info>
        <gp:usage-rules>
          <bp:retransmission-allowed>false
          </bp:retransmission-allowed>
        </gp:usage-rules>
      </gp:geopriv>
    </tns:status>
  </tns:tuple>
</tns:presence>
</tns:presence>
```

The following XML instance document is an example of a notification that encodes a PIDF-LO location in a GML 3.1 polygon geometry.

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:presence entity="pres:presentity@example.com"
  xmlns:p="http://www.w3.org/XML/1998/namespace"
  xmlns:tns="urn:ietf:params:xml:ns:pidf"
  xmlns:gp="urn:ietf:params:xml:ns:pidf:geopriv10"
  xmlns:bp="urn:ietf:params:xml:ns:pidf:geopriv10:basicPolicy"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <tns:tuple id="03783453">
    <tns:status>
      <gp:geopriv>
        <gp:location-info>
          <gml:extentOf>
            <gml:Polygon gml:id="C648AF9A54"
              srsName="urn:ogc:def:crs:EPSG::4326">
              <gml:name>myOffice</gml:name>
              <gml:exterior>
                <gml:LinearRing>
                  <gml:pos>48.222761 16.369345</gml:pos>
                  <gml:pos>48.222518 16.371018</gml:pos>
                  <gml:pos>48.221049 16.369951</gml:pos>
                  <gml:pos>48.220977 16.369135</gml:pos>
                  <gml:pos>48.222761 16.369345</gml:pos>
                </gml:LinearRing>
              </gml:exterior>
            </gml:Polygon>
          </gml:extentOf>
        </gp:location-info>
        <gp:usage-rules>
          <bp:retransmission-allowed>false
          </bp:retransmission-allowed>
        </gp:usage-rules>
      </gp:geopriv>
    </tns:status>
  </tns:tuple>
</tns:presence>
```

The following XML instance document is an example of a notification that encodes a PIDF-LO location as a set of containment status elements.

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:presence entity="pres:presentity@example.com"
  xmlns:p="http://www.w3.org/XML/1998/namespace"
  xmlns:tns="urn:ietf:params:xml:ns:pidf"
  xmlns:gp="urn:ietf:params:xml:ns:pidf:geopriv10"
  xmlns:bp="urn:ietf:params:xml:ns:pidf:geopriv10:basicPolicy"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:ct="urn:ietf:params:xml:ns:pidf:geopriv10:containment"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <tns:tuple id="03783453">
    <tns:status>
      <gp:geopriv>
        <gp:location-info>
          <ct:pidfResource>
            <ct:containment position="outside">
              <gml:extentOf>
                <gml:Polygon gml:id="74A4F34081"
                  srsName="urn:ogc:def:crs:EPSG::4326">
                  <gml:name>myOffice</gml:name>
                  <gml:exterior>
                    <gml:LinearRing>
                      <gml:pos>48.222761 16.369345</gml:pos>
                      <gml:pos>48.222518 16.371018</gml:pos>
                      <gml:pos>48.221049 16.369951</gml:pos>
                      <gml:pos>48.220977 16.369135</gml:pos>
                      <gml:pos>48.222761 16.369345</gml:pos>
                    </gml:LinearRing>
                  </gml:exterior>
                </gml:Polygon>
              </gml:extentOf>
            </ct:containment>
            <ct:containment position="inside">
              <gml:extentOf
                xlink:href="http://www.example.com/Vienna"/>
            </ct:containment>
          </ct:pidfResource>
        </gp:location-info>
        <gp:usage-rules>
          <bp:retransmission-allowed>false
          </bp:retransmission-allowed>
        </gp:usage-rules>
      </gp:geopriv>
    </tns:status>
  </tns:tuple>
</tns:presence>
```

The following XML instance document is an example of an empty notification.

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:presence entity="pres:presentity@example.com"
  xmlns:p="http://www.w3.org/XML/1998/namespace"
  xmlns:tns="urn:ietf:params:xml:ns:pidf"
  xmlns:gp="urn:ietf:params:xml:ns:pidf:geopriv10"
  xmlns:bp="urn:ietf:params:xml:ns:pidf:geopriv10:basicPolicy"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <tns:tuple id="03783453">
    <tns:status>
      <gp:geopriv>
        <gp:location-info>
        </gp:location-info>
        <gp:usage-rules>
        </gp:usage-rules>
      </gp:geopriv>
    </tns:status>
  </tns:tuple>
</tns:presence>
```

The following XML instance document is an example of a notification that aggregates location data from a GPS module in the terminal and the location of a wireless cell, where the terminal is booked in.

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:presence entity="pres:presentity@example.com"
  xmlns:p="http://www.w3.org/XML/1998/namespace"
  xmlns:tns="urn:ietf:params:xml:ns:pidf"
  xmlns:gp="urn:ietf:params:xml:ns:pidf:geopriv10"
  xmlns:bp="urn:ietf:params:xml:ns:pidf:geopriv10:basicPolicy"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <tns:tuple id="03783453">
    <tns:status>
      <gp:geopriv>
        <gp:location-info>
          <gml:position>
            <gml:Point srsName="urn:ogc:def:crs:EPSG::4326">
              <gml:pos>48.208481 16.372601</gml:pos>
            </gml:Point>
          </gml:position>
        </gp:location-info>
        <gp:method>GPS</gp:method>
        <gp:usage-rules>
          <bp:retransmission-allowed>false
          </bp:retransmission-allowed>
        </gp:usage-rules>
      </gp:geopriv>
    </tns:status>
  </tns:tuple>
  <tns:tuple id="87346740">
    <tns:status>
      <gp:geopriv>
        <gp:location-info>
          <gml:position>
            <gml:Point srsName="urn:ogc:def:crs:EPSG::4326">
              <gml:pos>48.207741 16.371378</gml:pos>
            </gml:Point>
          </gml:position>
        </gp:location-info>
        <gp:method>Cell</gp:method>
        <gp:usage-rules>
          <bp:retransmission-allowed>false
          </bp:retransmission-allowed>
        </gp:usage-rules>
      </gp:geopriv>
    </tns:status>
  </tns:tuple>
</tns:presence>
```

5.6. Notifier processing of SUBSCRIBE requests

The processing rules specified in section 6.6 of RFC 3856 [4], in particular regarding Authentication and Authorization, apply also for the location presence event package.

5.7. Notifier generation of NOTIFY requests

A notifier SHOULD check the subscriber's authorization, before sending location presence notifications.

The location presence event package specifies location filters , that MAY be used by the notifier to control the rate of notifications. It is the general idea of this document to reduce the number of NOTIFY messages to the necessary amount instead of using less scalable techniques like flooding or polling. If no filter document is passed with a subscription, the notifier SHOULD answer with at least one notification, containing the current position of the presentity. It is however the notifier's own decision when to send NOTIFY messages and what content to include in the NOTIFY body.

A Location Presence Agent (LPA) is co-located with the Presence User Agent (PUA) and therefore has direct access to any positioning mechanism that is integrated in the terminal that runs the PUA and the LPA (e.g. a GPS module). A LPA SHOULD be used to generate location presence events that use data from a terminal built-in positioning mechanism. Thus filter documents can effectively be used to reduce the rate of notifications.

A notifier that does not want to reveal its location to the subscriber MAY send an empty PIDF-LO 'location-info' element.

A notifier that wants to send a fake notification MAY send a 'containment' element with the containment status 'undefined'.

A notifier MAY send a partial containment notification, meaning that the containment state is not listed for all requested areas.

5.8. Subscriber processing of NOTIFY requests

This document does not specify any additional processing requirements for NOTIFY requests at the subscriber.

5.9. Handling of forked requests

Like RFC 3856 [4] this specification only allows a single dialog to be constructed as a result of emitting an initial SUBSCRIBE request. This guarantees that only a single PA is generating notifications for

a particular subscription to a particular presentity.

This specification defines a Location Presence Data Aggregator (LPDA) that acts as a Presence Agent (PA) on behalf of the presentity. The LPDA aggregates location presence data from different location data sources.

5.10. Rate of notifications

A location presence event notifier SHOULD NOT generate notifications for a single presentity at a rate of more than once every 5 seconds.

5.11. State Agents

A Presence Server as specified in RFC 3856 [4] is a PA that is not co-located with a PUA and acts as a state agent, by aggregating presence information from different PUA into one presence document.

This document specifies a Location Presence Data Aggregator (LPDA) that acts as a Presence Server for location presence data. The LPDA aggregates location presence data from different sources. A LPDA MUST only aggregate location data from different positioning methods and MUST indicate the positioning method in an PIDF-LO 'method' element.

An initial SUBSCRIBE request towards the LPDA will establish a dialog between the LPDA and the subscriber. If the LPDA wants to retrieve location presence data from a LPA, it has to establish a another dialog by sending a new SUBSCRIBE to the LPA. Section 6.1.1 'Aggregation, Authentication, and Authorization' for state agents in RFC 3856 [4] also applies for this document.

5.12. Use of URIs to Retrieve State

A 'xlink:href' attribute in a 'gml:extentOf' element MAY be used in subscription filters and in notifications, instead of giving a list of coordinates that form a polygon.

```
<gml:extentOf xlink:href="http://www.example.com/myOffice"/>
```

Using 'xlink:href' references reduces message size and may be used to increase privacy, if the given link is not publicly accessible. Furthermore the subscriber may not be interested in the exact location of the presentity, but rather in the logical containment state regarding the referenced area.

6. Usage of Presence URIs

Presence URIs shall be processed and used as specified in RFC 3856 [4].

7. Example Message Flow

The examples in this sections illustrate the usage of the location presence event package.

When the value of the Content-Length header field is "..." this means that the value should be whatever the computed length of the body is.

The following example flow shows a simple location presence fetch operation.

F1 SUBSCRIBE watcher -> LPA

```
SUBSCRIBE sip:marco@tuwien.ac.at SIP/2.0
Via: SIP/2.0/TCP watcherhost.a1.net;branch=z9hG4bK-8529-1-0
From: "Rudolf" <sip:rudolf@a1.net>;tag=1
To: "Marco" <sip:marco@tuwien.ac.at>
Call-ID: 1-8529@watcherhost.a1.net
CSeq: 1 SUBSCRIBE
Contact: sip:rudolf@watcherhost.a1.net
Max-Forwards: 70
Subject: Location Presence Test
Event: locpres
Accept: application/location-presence-pidf+xml
Expires: 0
Content-Length: 0
```

F2 200 OK LPA -> watcher

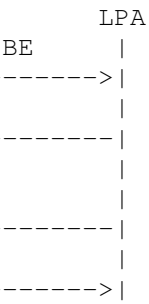
```
SIP/2.0 200 OK
Via: SIP/2.0/TCP watcherhost.a1.net;branch=z9hG4bK-8529-1-0
From:  "Rudolf" <sip:rudolf@a1.net>;tag=1
To:  "Marco" <sip:marco@tuwien.ac.at>;tag=1
Call-ID: 1-8529@watcherhost.a1.net
CSeq: 1 SUBSCRIBE
Event: locpres
Expires: 0
Contact: <sip:tuwien.ac.at;transport=TCP>
Content-Length: 0

F3 NOTIFY LPA -> watcher

NOTIFY sip:rudolf@watcherhost.a1.net SIP/2.0
Via: SIP/2.0/TCP tuwien.ac.at
From:  "Marco" <sip:marco@tuwien.ac.at>;tag=1
To:  "Rudolf" <sip:rudolf@a1.net>;tag=1
Call-ID: 1-8529@watcherhost.a1.net
CSeq: 1 NOTIFY
Event: locpres
Subscription-State: terminated;reason=timeout
Contact: <sip:tuwien.ac.at;transport=TCP>
Content-Type: application/location-presence-pidf+xml
Content-Length: ...

<?xml version="1.0" encoding="UTF-8"?>
<tns:presence entity="pres:presentity@example.com"
  xmlns:p="http://www.w3.org/XML/1998/namespace"
  xmlns:tns="urn:ietf:params:xml:ns:pidf"
  xmlns:gp="urn:ietf:params:xml:ns:pidf:geopriv10"
  xmlns:bp="urn:ietf:params:xml:ns:pidf:geopriv10:basicPolicy"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >

  <tns:tuple id="03783453">
    <tns:status>
      <gp:geopriv>
        <gp:location-info>
          <gml:position>
            <gml:Point srsName="urn:ogc:def:crs:EPSG::4326">
              <gml:pos>48.208481 16.372601</gml:pos>
            </gml:Point>
          </gml:position>
        </gp:location-info>
        <gp:usage-rules>
          <bp:retransmission-allowed>false
          </bp:retransmission-allowed>
        </gp:usage-rules>
```

```
          <gp:method>GPS</gp:method>
        </gp:geopriv>
      </tns:status>
    </tns:tuple>
  </tns:presence>
```

F4 200 OK watcher -> LPA

```
SIP/2.0 200 OK
Via: SIP/2.0/TCP tuwien.ac.at
From:  "Marco" <sip:marco@tuwien.ac.at>;tag=1
To:  "Rudolf" <sip:rudolf@a1.net>;tag=1
Call-ID: 1-8529@watcherhost.a1.net
CSeq: 1 NOTIFY
Content-Length: 0
```

The following example message flow shows a subscription for location presence containment state. Notice that the actual geometry of the watched region is only refered to by a URL.



F1 SUBSCRIBE watcher -> LPA

```
SUBSCRIBE sip:marco@tuwien.ac.at SIP/2.0
Via: SIP/2.0/TCP watcherhost.a1.net;branch=z9hG4bK-22943-1-0
From: "Rudolf" <sip:rudolf@a1.net>;tag=1
To: "Marco" <sip:marco@tuwien.ac.at>
Call-ID: 1-22943@watcherhost.a1.net
CSeq: 1 SUBSCRIBE
Contact: sip:rudolf@watcherhost.a1.net
Max-Forwards: 70
Subject: Location Presence Test
Event: locpres
Accept: application/location-presence-pidf+xml
Expires: 0
```

```
   Content-Length: ...

   <?xml version="1.0" encoding="UTF-8"?>
   <p:location-filter xmlns:gml="http://www.opengis.net/gml"
     xmlns:p="urn:ietf:params:xml:ns:location-presence-filter"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:xlink="http://www.w3.org/1999/xlink">

     <enterOrExit>
       <gml:extentOf
       xlink:href="http://www.a1.net/userProfiles/A9D82F6987"/>
     </enterOrExit>
   </p:location-filter>


   F2 200 OK LPA -> watcher

   SIP/2.0 200 OK
   Via: SIP/2.0/TCP watcherhost.a1.net;branch=z9hG4bK-22943-1-0
   From:  "Rudolf" <sip:rudolf@a1.net>;tag=1
   To:  "Marco" <sip:marco@tuwien.ac.at>;tag=1
   Call-ID: 1-22943@watcherhost.a1.net
   CSeq: 1 SUBSCRIBE
   Event: locpres
   Expires: 0
   Contact: <sip:tuwien.ac.at;transport=TCP>
   Content-Length: 0

   F3 NOTIFY LPA -> watcher

   NOTIFY sip:rudolf@watcherhost.a1.net SIP/2.0
   Via: SIP/2.0/TCP tuwien.ac.at
   From:  "Marco" <sip:marco@tuwien.ac.at>;tag=1
   To:  "Rudolf" <sip:rudolf@a1.net>;tag=1
   Call-ID: 1-22943@watcherhost.a1.net
   CSeq: 1 NOTIFY
   Event: locpres
   Subscription-State: terminated;reason=timeout
   Contact: <sip:tuwien.ac.at;transport=TCP>
   Content-Type: application/location-presence-pidf+xml
   Content-Length: ...

   <?xml version="1.0" encoding="UTF-8"?>
   <tns:presence entity="pres:presentity@example.com"
     xmlns:p="http://www.w3.org/XML/1998/namespace"
     xmlns:tns="urn:ietf:params:xml:ns:pidf"
     xmlns:gp="urn:ietf:params:xml:ns:pidf:geopriv10"
     xmlns:bp="urn:ietf:params:xml:ns:pidf:geopriv10:basicPolicy"
```

```
     xmlns:gml="http://www.opengis.net/gml"
     xmlns:ct="urn:ietf:params:xml:ns:pidf:geopriv10:containment"
     xmlns:xlink="http://www.w3.org/1999/xlink"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

     <tns:tuple id="03783453">
       <tns:status>
         <gp:geopriv>
           <gp:location-info>
             <ct:pidfResource>
               <ct:containment position="inside">
                 <gml:extentOf
             xlink:href="http://www.a1.net/userProfiles/A9D82F6987"/>
               </ct:containment>
             </ct:pidfResource>
           </gp:location-info>
           <gp:usage-rules>
             <bp:retransmission-allowed>false
             </bp:retransmission-allowed>
           </gp:usage-rules>
         </gp:geopriv>
       </tns:status>
     </tns:tuple>
   </tns:presence>

   F4 200 OK watcher -> LPA

   SIP/2.0 200 OK
   Via: SIP/2.0/TCP tuwien.ac.at
   From:  "Marco" <sip:marco@tuwien.ac.at>;tag=1
   To:  "Rudolf" <sip:rudolf@a1.net>;tag=1
   Call-ID: 1-22943@watcherhost.a1.net
   CSeq: 1 NOTIFY
   Content-Length: 0
```

8. Security Considerations

Location information provides considerable value to information and communication services. On the other hand, users are concerned about revealing their position data to others, especially to un-trusted third party applications. Furthermore, most countries have legal restrictions that regulate processing of personal data and the protection of privacy in electronic communications. It is of utmost importance that the users can control who gets access to their location data and that the transport in the network of such sensitive data is protected by strong security mechanisms.

These security and privacy considerations are covered by the general presence specification and any implementation of this document MUST follow the security considerations in RFC 3856 [4].

9. IANA Considerations

9.1. Location Presence Event Package

This specification registers an event package, based on the registration procedures defined in RFC 3265 [3]. The following is the information required for such a registration:

Package Name: locpres

Published Document: please assign

Person to Contact: Rudolf Pailer, r.pailer@a1.net

9.2. MIME Registration for application/location-presence-pidf+xml

MIME media type name: application

MIME subtype name: application/location-presence-pidf+xml

Required parameters: none.

Optional parameters: none.

Encoding considerations: Same as for XML.

Security considerations: see Section 8

Applications which use this media: The application/ location-presence-pidf+xml application subtype supports the exchange of location information inside presence documents.

Additional Information:

1. Magic number(s): N/A

2. File extension(s): N/A

3. Macintosh file type code: N/A

9.3. MIME Registration for application/ location-presence-delta-filter+xml

[Note to the editor: this application type is an updated version of application/location-delta-filter+xml defined in [19]].

MIME media type name: application

MIME subtype name: application/location-presence-delta-filter+xml

Required parameters: none.

Optional parameters: none.

Encoding considerations: Same as for XML.

Security considerations: see Section 8

Applications which use this media: The application/ location-presence-delta-filter+xml application subtype supports the exchange of filters to throttle asynchronous notifications of location information.

Additional Information:

1. Magic number(s): N/A

2. File extension(s): N/A

3. Macintosh file type code: N/A

9.4. URN Sub-Namespace Registration for urn:ietf:params:xml:ns:location-presence-filter

[Note to the editor: this schema URN refers to an updated version of the filter schema defined in [19]].

This section registers a new XML namespace, as per the guidelines in [12].

URI: The URI for this namespace is urn:ietf:params:xml:ns:location-presence-filter.

Registrant Contact: IETF, GEOPRIV working group, <geopriv@ietf.org>, as delegated by the IESG <iesg@ietf.org>.

XML:

```
BEGIN
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN"
          "http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta http-equiv="content-type"
     content="text/html;charset=iso-8859-1"/>
  <title>Location Presence Filter Namespace</title>
</head>
<body>
  <h1>Namespace for PIDF-LO Location Presence Filters</h1>
  <h2>urn:ietf:params:xml:ns:location-presence-filter</h2>
  <p>See <a href="[[[URL of published RFC]]]">RFCXXXX</a>.</p>
</body>
</html>
END
```

9.5. Schema Registration For location-presence-filter

[Note to the editor: this schema is an updated version of the filter schema defined in [19]].

This specification registers a schema, as per the guidelines in [12].

URI: please assign

Registrant Contact: IETF, GEOPRIV working group, <geopriv@ietf.org>, as delegated by the IESG <iesg@ietf.org>.

XML: see Appendix A.1.1

9.6. URN Sub-Namespace Registration for urn:ietf:params:xml:ns:pidf:geopriv10:containment

[Note to the editor: this schema URN was originally defined in [19]].

This section registers a new XML namespace, as per the guidelines in [12].

URI: The URI for this namespace is urn:ietf:params:xml:ns:pidf:geopriv10:containment.

Registrant Contact: IETF, GEOPRIV working group,
<geopriv@ietf.org>, as delegated by the IESG <iesg@ietf.org>.

XML:

```
BEGIN
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN"
          "http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta http-equiv="content-type"
     content="text/html;charset=iso-8859-1"/>
  <title>PIDF-LO Location Containment Namespace</title>
</head>
<body>
  <h1>Namespace for PIDF-LO location containment elements</h1>
  <h2>urn:ietf:params:xml:ns:pidf:geopriv10:containment</h2>
  <p>See <a href="[[[URL of published RFC]]]">RFCXXXX</a>.</p>
</body>
</html>
END
```

9.7. Schema Registration For containment

[Note to the editor: this schema was originally defined in [19]].

This specification registers a schema, as per the guidelines in [12].

URI: please assign

Registrant Contact: IETF, GEOPRIV working group,
<geopriv@ietf.org>, as delegated by the IESG <iesg@ietf.org>.

XML: see Appendix A.2.1

10. Contributors

The following people have contributed to this work:

Florian WEGSCHEIDER
mobilkom austria AG
Obere Doneustrasse 29
A-1020 Vienna, Austria
mailto:f.wegscheider@mobilkom.at

Sandford BESSLER

Telecommunications Research Center Vienna (ftw.)
Obere Donaustrasse 29
Donau City Strasse 1
A-1220 Vienna, Austria
mailto:bessler@ftw.at

Joachim FABINI
Institute of Broadband Communications
Vienna University of Technology
Favoritenstrasse 9/E388
A-1040 Vienna, Austria
mailto:Joachim.Fabini@tuwien.ac.at

Marco HAPPENHOFER
Institute of Broadband Communications
Vienna University of Technology
Favoritenstrasse 9/E388
A-1040 Vienna, Austria
mailto:Marco.Happenhofer@tuwien.ac.at

11. Acknowledgements

Part of this work has been performed within the project "SIMS-Services in the IP Multimedia Subsystem" at the Telecommunications Research Center Vienna (http://www.ftw.at) and has been funded in the framework of the Austrian Kplus Competence Center Programme.

12. References

12.1. Normative References

[1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[2] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[3] Roach, A., "Session Initiation Protocol (SIP)-Specific Event Notification", RFC 3265, June 2002.

[4] Rosenberg, J., "A Presence Event Package for the Session Initiation Protocol (SIP)", RFC 3856, August 2004.

[5] Peterson, J., "A Presence Architecture for the Distribution of GEOPRIV Location Objects", RFC 4079, July 2005.

[6] Schulzrinne, H., Gurbani, V., Kyzivat, P., and J. Rosenberg, "RPID: Rich Presence Extensions to the Presence Information Data Format (PIDF)", RFC 4480, July 2006.

[7] 3GPP, "Location Services (LCS); Service description; Stage 1", 3GPP TS 22.071 3.5.0, March 2004.

[8] Peterson, J., "Common Profile for Presence (CPP)", RFC 3859, August 2004.

[9] Day, M., Rosenberg, J., and H. Sugano, "A Model for Presence and Instant Messaging", RFC 2778, February 2000.

[10] Peterson, J., "A Presence-based GEOPRIV Location Object Format", RFC 4119, December 2005.

[11] "OpenGIS Geography Markup Language (GML) Implementation Specification, OGC 03-105r1", 02 2004.

[12] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

[13] Sugano, H., Fujimoto, S., Klyne, G., Bateman, A., Carr, W., and J. Peterson, "Presence Information Data Format (PIDF)", RFC 3863, August 2004.

12.2. Informative References

[14] "OpenGIS Geography Markup Language (GML) Encoding Specification, OGC 02-023r4", 12 2002.

[15] OGC, "Definition identifier URNs in OGC namespace, OGC Best Practices Paper 06-023r1", 08 2006.

[16] EPSG, "EPSG Geodetic Parameter Dataset, version 6.11_2", 10 2006.

[17] Thomson, M., "Geodetic Shapes for the Representation of Uncertainty in PIDF-LO", draft-thomson-geopriv-geo-shape-03 (work in progress), December 2006.

[18] Tschofenig, H., "GEOPRIV PIDF-LO Usage Clarification, Considerations and Recommendations", draft-ietf-geopriv-pdif-lo-profile-05 (work in progress), October 2006.

[19] Mahy, R., "A Document Format for Filtering and Reporting Location Notications in PIDF-LO",

draft-mahy-geopriv-loc-filters-01 (work in progress), March 2006.

Appendix A. Filtering and Reporting Location Notifications in PIDF-LO documents (Normative)

This specification uses XML schemas proposed in the Internet Draft [19] that has expired in September 2006, but can still be seen at 'ht tp://www3.ietf.org/proceedings/06jul/IDs/ draft-ietf-geopriv-loc-filters-00.txt'. In order to remove the dependency on an expired draft, this appendix lists the used XML schemas.

A.1. Location Presence Filter Format

The granularity of notifications necessary for various geographic location applications varies dramatically. The subscriber should be able to get asynchronous notifications with appropriate granularity and accuracy, without having to poll or flood the network with notifications which are not important to the application. Notifications should only happen when the notification would be considered an interesting event to the subscriber. This section defines an XML filter format to describe interesting conditions or events.

This document also defines a MIME type for this location filter format: application/location-presence-filter+xml.

This document defines the following as an initial list of Interesting Events:

1. the resource moves more than a specific distance horizontally or vertically since the last notification

2. the resource exceeds a specific speed

3. the resource enters or exits one or more GML objects (for example, a polygon) included or referenced in the filter.

The filter format starts with a top-level XML element called "<location-filter>", which contains one or more filter events. The semantics of multiple elements inside a location-filter is a logical OR. In other words, if any of the individual filter events occurs, the event satisfies the location-filter and triggers a notification.

The "movedHoriz" and "movedVert" filter events each indicate a minimum horizontal motion or vertical distance (respectively) that

the resource must have moved from the location of the resource when the last notification was sent in order to trigger this event. The default unit of measurement for these events is meters.

Similarly, the "speedExceeds" filter event indicates a minimum horizontal speed of the resource before the "speedExceeds" event is triggered. The default unit of measurement for these events is meters per second.

The "valueChanges" filter event contains a string which is interpreted as an XPath expression evaluated within the context of the location-info element of the PIDF-LO document which would be generated by the notification. The XPath expression MUST evaluate to only a single Xpath node.

Finally, the "enterOrExit" filter event is satisfied when the resource enters or exits a specified location. The original draft types the "enterOrExit" element as a GML feature. This was considered to be too restrictive and complicated, as it requires from the subscriber and the notifier to agree on a not further specified GML application schema. The "enterOrExit" element in this document may contain any XML element. For a subscription with the content-type 'application/location-presence-filter+xml', the "enterOrExit" element MUST contain a 'gml:extentOf' element. In most cases subscribers that use location filters based on "enterOrExit" events are especially interested in the resource's relationship to those locations. Consequently, the notifier SHOULD include a "containment" element for each location mentioned in the location-filter which has changed its containment properties with respect to the resource since the last notification.

A.1.1. Location Presence Filter Schema

The following XML document defines the location presence filter schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  targetNamespace="urn:ietf:params:xml:ns:location-presence-filter"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:gml="http://www.opengis.net/gml">

  <xs:element name="location-filter">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="movedHoriz" type="gml:MeasureType"
          minOccurs="0" maxOccurs="1" />
        <xs:element name="movedVert" type="gml:MeasureType"
          minOccurs="0" maxOccurs="1" />
        <xs:element name="speedExceeds" type="gml:MeasureType"
          minOccurs="0" maxOccurs="1" />

        <!-- this type needs to hold an XPath statement -->
        <xs:element name="valueChanges" type="xs:string"
          minOccurs="0" maxOccurs="unbounded" />

        <!--
          The original enterOrExit element from
          draft-mahy-geopriv-loc-filters-01 requires the use
          of an application XML document, extending GML.
          An more general alternative is specified here.
          <xs:element name="enterOrExit"
            type="gml:FeaturePropertyType"
            minOccurs="0" maxOccurs="unbounded" />
        -->
        <xs:element name="enterOrExit"
          minOccurs="0"
          maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:any namespace="##other" processContents="lax"
                minOccurs="0" maxOccurs="unbounded" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>

        <!-- Do we want to include this to allow new filters? -->
        <xs:any namespace="##other" processContents="lax"
          minOccurs="0" maxOccurs="unbounded" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

A.2. Containment

This section defines a schema for describing the resource's location relative to a region or list of regions which might contain the resource. (These regions can be defined dynamically in an "enterOrExit" element in a subscription filter, or defined on the notifier using some out-of-band mechanism.) The "pidfResource" element is placed inside the location-info element in a PIDF-LO document. The pidfResource element can contain zero or more "containment" elements. Each containment element has a GML Feature sub-element (of type "FeaturePropertyType") and a mandatory attribute which specifies if the PIDF resource is inside or outside of the feature, or if the position of the resource with respect to the region or region list is undefined.

If the subscriber is not authorized to know the relative position, the notifier MUST NOT reveal this private information. The RECOMMENDED way to prevent the subscriber from seeing private location data of this type is to return a containment element whose position attribute is "undefined". Note that in some cases, the containment information may be more interesting than the actual raw location. It is not necessary to convey a concrete geo location in a PIDF-LO if the subscriber is only interested in or authorized to see the containment status.

A.2.1. Containment Schema

The following XML document defines the containment schema.

```
<?xml version="1.0" encoding="UTF-8"?>
   <xs:schema
   targetNamespace="urn:ietf:params:xml:ns:pidf:geopriv10:containment"
       xmlns:xs="http://www.w3.org/2001/XMLSchema"
       xmlns:gml="http://www.opengis.net/gml"
   xmlns:pr="urn:ietf:params:xml:ns:pidf:geopriv10:containment" >
     <xs:element name="pidfResource">
       <xs:complexType>
           <xs:sequence>
               <xs:element ref="pr:containment"
                minOccurs="0" maxOccurs="unbounded"/>
           </xs:sequence>
       </xs:complexType>
     </xs:element>
     <xs:element name="containment">
       <xs:complexType>
           <xs:sequence>
             <xs:any namespace="http://www.opengis.net/gml"
                   minOccurs="1" maxOccurs="1"/>
           </xs:sequence>
         <xs:attribute name="position" use="required">
            <xs:simpleType>
           <xs:restriction base="xs:string">
               <xs:enumeration value="inside"></xs:enumeration>
               <xs:enumeration value="outside"></xs:enumeration>
               <xs:enumeration value="undefined"></xs:enumeration>
             </xs:restriction>
           </xs:simpleType>
         </xs:attribute>
       </xs:complexType>
     </xs:element>
   </xs:schema>
```

Author's Address

Rudolf Pailer
mobilkom austria AG
Obere Donaustrasse 29
Vienna A-1020
Austria

Phone: +43-664-3316983
Email: r.pailer@a1.net

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).

Appendix B: Program Output

This appendix shows the program output for the calculation of the steady state probabilities and performance measures of the model presented in section 5.3.1 *System with Two Queues*. The following program output shows how the state indices i, j, k are mapped to a single index and then to a valid state index. State S(2,1,1) for example has the single index 22 and the valid index 15. The valid indices are used in the matrix that is solved.

```
State(0) = ValidState(0) = State (0,0,0):    μ * P(0,0,0) + μ * P(0,0,0) = l1 *
P(0,0,1) + l2 * P(0,0,2)
State(1) = ValidState(1) = State (0,0,1):    l1 * P(0,0,1) + μ * P(0,0,1) + μ *
P(0,0,1) = l1 * P(1,0,1) + l2 * P(0,1,1) + μ * P(0,0,0)
State(2) = ValidState(2) = State (0,0,2):    l2 * P(0,0,2) + μ * P(0,0,2) + μ *
P(0,0,2) = l1 * P(1,0,2) + l2 * P(0,1,2) + μ * P(0,0,0)
State(4) = ValidState(3) = State (0,1,1):    l2 * P(0,1,1) + qμ * P(0,1,1) + qμ *
P(0,1,1) = l1 * P(1,1,1) + l2 * P(0,2,1) + μ * P(0,0,2)
State(5) = ValidState(4) = State (0,1,2):    l2 * P(0,1,2) + μ * P(0,1,2) + μ *
P(0,1,2) = l1 * P(1,1,2) + l2 * P(0,2,2) + μ * P(0,0,2)
State(7) = ValidState(5) = State (0,2,1):    l2 * P(0,2,1) + qμ * P(0,2,1) + qμ *
P(0,2,1) = l1 * P(1,2,1) + μ * P(0,1,2)
State(8) = ValidState(6) = State (0,2,2):    l2 * P(0,2,2) = l1 * P(1,2,2) + μ *
P(0,1,2)
State(10) = ValidState(7) = State (1,0,1):    l1 * P(1,0,1) + μ * P(1,0,1) + μ *
P(1,0,1) = l1 * P(2,0,1) + l2 * P(1,1,1) + μ * P(0,0,1)
State(11) = ValidState(8) = State (1,0,2):    l1 * P(1,0,2) + (1-q)μ * P(1,0,2) +
(1-q)μ * P(1,0,2) = l1 * P(2,0,2) + l2 * P(1,1,2) + μ * P(0,0,1)
State(13) = ValidState(9) = State (1,1,1):    l1 * P(1,1,1) + l2 * P(1,1,1) + qμ *
P(1,1,1) + qμ * P(1,1,1) = l1 * P(2,1,1) + l2 * P(1,2,1) + qμ * P(0,1,1) + (1-q)μ *
P(1,0,2)
State(14) = ValidState(10) = State (1,1,2):    l1 * P(1,1,2) + l2 * P(1,1,2) + (1-
q)μ * P(1,1,2) + (1-q)μ * P(1,1,2) = l1 * P(2,1,2) + l2 * P(1,2,2) + qμ * P(0,1,1)
+ (1-q)μ * P(1,0,2)
State(16) = ValidState(11) = State (1,2,1):    l1 * P(1,2,1) + l2 * P(1,2,1) + qμ *
P(1,2,1) + qμ * P(1,2,1) = l1 * P(2,2,1) + qμ * P(0,2,1) + (1-q)μ * P(1,1,2)
State(17) = ValidState(12) = State (1,2,2):    l1 * P(1,2,2) + l2 * P(1,2,2) = l1 *
P(2,2,2) + qμ * P(0,2,1) + (1-q)μ * P(1,1,2)
State(19) = ValidState(13) = State (2,0,1):    l1 * P(2,0,1) = l2 * P(2,1,1) + μ *
P(1,0,1)
State(20) = ValidState(14) = State (2,0,2):    l1 * P(2,0,2) + (1-q)μ * P(2,0,2) +
(1-q)μ * P(2,0,2) = l2 * P(2,1,2) + μ * P(1,0,1)
State(22) = ValidState(15) = State (2,1,1):    l1 * P(2,1,1) + l2 * P(2,1,1) = l2 *
P(2,2,1) + qμ * P(1,1,1) + (1-q)μ * P(2,0,2)
State(23) = ValidState(16) = State (2,1,2):    l1 * P(2,1,2) + l2 * P(2,1,2) + (1-
q)μ * P(2,1,2) + (1-q)μ * P(2,1,2) = l2 * P(2,2,2) + qμ * P(1,1,1) + (1-q)μ *
P(2,0,2)
State(25) = ValidState(17) = State (2,2,1):    l1 * P(2,2,1) + l2 * P(2,2,1) = qμ *
P(1,2,1) + (1-q)μ * P(2,1,2)
State(26) = ValidState(18) = State (2,2,2):    l1 * P(2,2,2) + l2 * P(2,2,2) = qμ *
P(1,2,1) + (1-q)μ * P(2,1,2)
```

The resulting coefficients of matrix A (using the valid state indices) are:

```
-6   8   8   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
 2 -14   0   8   8   0   0   0   0   0   0   0   0   0   0   0   0   0   0
 4   0 -14   0   0   0   0   8   8   0   0   0   0   0   0   0   0   0   0
 0   2   0 -14   0   8   8   0   0   0   0   0   0   0   0   0   0   0   0
 0   0   2   0 -14   0   0   0   0   1.6   1.6   0   0   0   0   0   0   0   0
 0   0   0   2   0 -12   0   0   0   0   0   0   0   0   0   0   0   0   0
 0   0   0   0   2   0 -12   0   0   0   0   1.6   1.6   0   0   0   0   0   0
 0   4   0   0   0   0   0 -14   0   6.4   6.4   0   0   0   0   0   0   0   0
 0   0   4   0   0   0   0   0 -14   0   0   0   0   8   8   0   0   0   0
```

```
 0   0   0   4   0   0   0   2   0 -14   0   6.4   6.4   0   0   0   0  0   0
 0   0   0   0   4   0   0   0   2   0 -14   0   0   0   0   1.6   1.6  0   0
 0   0   0   0   0   4   0   0   0   2   0 -12   0   0   0   0   0   0
 0   0   0   0   0   0   4   0   0   0   2   0 -12   0   0   0   0  1.6  1.6
 0   0   0   0   0   0   0   4   0   0   0   0   0 -10   0   6.4   6.4  0   0
 0   0   0   0   0   0   0   0   4   0   0   0   0   0 -10   0   0   0   0
 0   0   0   0   0   0   0   0   0   4   0   0   0   2   0 -10   0  6.4  6.4
 0   0   0   0   0   0   0   0   0   0   4   0   0   0   2   0 -10   0   0
 0   0   0   0   0   0   0   0   0   0   0   4   0   0   0   2   0 -8   0
 0   0   0   0   0   0   0   0   0   0   0   0   4   0   0   0   2  0  -8
```

The last row of Matrix A is then replaced by the probability normalization condition:

```
6   8   8   0   0   0   0   0   0   0   0   0   0   0   0   0   0  0  0
 2 -14   0   8   8   0   0   0   0   0   0   0   0   0   0   0   0  0  0
 4   0 -14   0   0   0   0   8   8   0   0   0   0   0   0   0   0  0  0
 0   2   0 -14   0   8   8   0   0   0   0   0   0   0   0   0   0  0  0
 0   0   2   0 -14   0   0   0   0   1.6   1.6   0   0   0   0   0   0  0  0
 0   0   0   2   0 -12   0   0   0   0   0   0   0   0   0   0   0  0  0
 0   0   0   0   2   0 -12   0   0   0   0   1.6   1.6   0   0   0   0  0  0
 0   4   0   0   0   0   0 -14   0   6.4   6.4   0   0   0   0   0   0  0  0
 0   0   4   0   0   0   0   0 -14   0   0   0   0   8   8   0   0  0  0
 0   0   0   4   0   0   0   2   0 -14   0   6.4   6.4   0   0   0   0  0  0
 0   0   0   0   4   0   0   0   2   0 -14   0   0   0   0   1.6   1.6  0  0
 0   0   0   0   0   4   0   0   0   2   0 -12   0   0   0   0   0   0  0
 0   0   0   0   0   0   4   0   0   0   2   0 -12   0   0   0   0  1.6 1.6
 0   0   0   0   0   0   0   4   0   0   0   0   0 -10   0   6.4   6.4  0  0
 0   0   0   0   0   0   0   0   4   0   0   0   0   0 -10   0   0   0  0
 0   0   0   0   0   0   0   0   0   4   0   0   0   2   0 -10   0  6.4 6.4
 0   0   0   0   0   0   0   0   0   0   4   0   0   0   2   0 -10   0  0
 0   0   0   0   0   0   0   0   0   0   0   4   0   0   0   2   0 -8   0
 1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1  1  1
```

The system **Ax=b** is the solved with $\mathbf{b}^T$:

```
  0   0   0   0     0   0    0   0  0     0     0    0    0  0  0    0  0    0 1
```

Solving this system of linear equations by LU-decompositions gives the following steady state probabilities:

```
P((0,0)(0)) = P(StateValid( 0)) = P(State( 0)) = 0.33685533360234615
P((0,0)(1)) = P(StateValid( 1)) = P(State( 1)) = 0.07481054524112882
P((0,0)(2)) = P(StateValid( 2)) = P(State( 2)) = 0.1778309549606308
P((1,0)(1)) = P(StateValid( 3)) = P(State( 4)) = 0.016138174102549124
P((1,0)(2)) = P(StateValid( 4)) = P(State( 5)) = 0.030566446668839776
P((2,0)(1)) = P(StateValid( 5)) = P(State( 7)) = 0.002689695683758186
P((2,0)(2)) = P(StateValid( 6)) = P(State( 8)) = 0.006849472685420573
P((0,1)(1)) = P(StateValid( 7)) = P(State(10)) = 0.04202253962389262
P((0,1)(2)) = P(StateValid( 8)) = P(State(11)) = 0.1007539647560382
P((1,1)(1)) = P(StateValid( 9)) = P(State(13)) = 0.016631492229103643
P((1,1)(2)) = P(StateValid(10)) = P(State(14)) = 0.02853622242245595
P((2,1)(1)) = P(StateValid(11)) = P(State(16)) = 0.0036684805994366696
P((2,1)(2)) = P(StateValid(12)) = P(State(17)) = 0.009494506205167905
P((0,2)(1)) = P(StateValid(13)) = P(State(19)) = 0.04710237494033617
P((0,2)(2)) = P(StateValid(14)) = P(State(20)) = 0.04030158590241528
P((1,2)(1)) = P(StateValid(15)) = P(State(22)) = 0.027858567429876958
P((1,2)(2)) = P(StateValid(16)) = P(State(23)) = 0.019474806149465437
P((2,2)(1)) = P(StateValid(17)) = P(State(25)) = 0.008798882157187575
P((2,2)(2)) = P(StateValid(18)) = P(State(26)) = 0.00961595463995034
```