

MASTER'S THESIS

TECHNISCHE UNIVERSITÄT WIEN

INSTITUTE OF TELECOMMUNICATIONS

Deriving a Network Perspective of Cellular Mobile Networks Based on Crowd Sourced Benchmark Tests

Author:

Vaclav RAIDA

Supervisors:

Univ.Prof. Dipl.-Ing. Dr.techn. Markus RUPP
Projektass. Dipl.-Ing. Dr.techn. Philipp SVOBODA

November 9, 2017



Declaration of Authorship

I hereby declare that this thesis has been composed by me and is based on my own work, unless stated otherwise. No other person's work has been used without acknowledgment in this thesis. All references have been quoted, and all sources of information, including figures, have been specifically acknowledged.

Date:

Signature:

Abstract

The concept of crowdsourced measurements allows service providers to outsource the expensive and time consuming task of performance measurements to the end user device. This approach is obviously appealing for the industry, though giving up the control on the experiment in favor. In this thesis we took a first step into the analysis of crowdsourced data sets in order to build up a view of the current status of the network. This work aims at answering the question whether it is possible to use crowdsourced data to meaningfully characterize properties of the network.

The thesis i) introduces theoretical concepts for systematic description and processing of data rate series based on data volume samples; ii) describes implementation of generic framework allowing for repeated measurements and benchmarking various measurement tools; iii) analyzes results of controlled measurements produced by the same tools which are used for crowdsourced measurements; iv) analyzes results of crowdsourced measurements.

Controlled measurements reveal great potential of crowdsourced open data, yielding positive answer to the stated question. The use of real data set collected by the regulatory body in Austria, RTR, proved to be challenging due to multiple factors:

The first element is the pollution of the data set by systematic events, e.g. operator tests at special locations, operator optimizations in user profiling and many more. The second element is the dynamic of the data set in temporal and spatial dimension. The analysis and the reference measurements reveal clear time of day effects, e.g. clear diurnal load cycles in the cells.

Finally, current implementation of RTR data is recorded in a lossy fashion. The analysis shows that the information loss cannot be recovered. Therefore, limitations due to user tariffs and network dynamics cannot be precisely removed.

However, this work shows that even considering all these effects it is possible to already use the data set to gather network performance benchmark figures.

Contents

Introduction	1
1 Benchmarking Methodology for Mobile Cellular Networks	3
1.1 Introduction	3
1.2 Terminology	3
1.3 LTE	4
1.3.1 Physical Layer Measurements at UE	4
1.4 TCP	5
1.4.1 Round-Trip Time	5
1.4.2 Bandwidth-Delay Product	5
1.4.3 Flow Control	6
1.4.4 Congestion Control	6
1.4.5 Overview of Congestion Avoidance Algorithms	9
1.5 Active Performance Measurements in Standardization	10
1.5.1 IETF – Framework for TCP Throughput Testing	10
1.5.2 3GPP – UE Application Layer Data Throughput Performance	12
1.6 First Analysis: TCP Related Effects	13
1.6.1 Measurement Setup	13
1.6.2 Flow Control Limitation	13
1.6.3 Impact of Bufferbloat	14
1.6.4 Congestion Control Limitation	15
1.7 Comparison of iPerf3, HTTP and FTP Throughput	19
1.7.1 Setup	20
1.7.2 Results	20
2 On the Connection of Data Volumes and Rates in Networks	21
2.1 From Volume to Rate	21
2.1.1 Basic Definitions	21
2.1.2 Resampling	23
2.1.3 Multiple Connections: Merging	25
2.1.4 Resampling and Merging Combined	27
2.1.5 Summary	27
2.2 Smoothing: Reducing Uncertainty of Binned Data	29
2.2.1 Definitions	29
2.2.2 Worst Case Uncertainty	30
2.2.3 Smoothing	30
2.3 Thinning	33
2.3.1 Thinning Algorithm	34
2.3.2 Implementation of Thinning Algorithm	34
2.3.3 Remarks	35
2.4 Oscillations	37

2.4.1	Model	37
2.4.2	Spectrogram	37
2.4.3	Reducing Oscillations	38
2.5	Extracting Network Performance From User Tests	40
2.5.1	Traffic Shaping Detectors (Existing Method)	40
2.5.2	Modified Traffic Shaping Detector	41
3	Measurement Framework and Tools	42
3.1	The Measurement Framework (CMPT)	42
3.1.1	Android Application	43
3.1.2	Web Interface	49
3.1.3	Server	49
3.1.4	Tools of Third Parties	50
3.1.5	Examples of Use Cases	50
3.1.6	Outlook and Limitations	51
3.2	RMBT	52
3.2.1	Test procedure	52
3.2.2	Thinning	54
3.2.3	More Control: Open-RMBT TU.2.2.12	55
3.2.4	Open-RMBT Applications in Different Countries	55
4	Evaluation of Controlled Measurements	56
4.1	Simplified Notation	56
4.2	Presence of Data Rate Oscillations	56
4.2.1	Measurement Setup 1	57
4.2.2	Measurement Setup 2	57
4.2.3	Conclusion and Possible Cause of Offset	61
4.3	Systematic Removing of Oscillations	62
4.3.1	Automatized Measurements	62
4.3.2	Minimizing MSD with Respect to Reference Signal	63
4.3.3	Suppressing Oscillations Without Knowing Reference	63
4.3.4	Numerical Evaluation	64
4.4	Oscillations and Thinning Combined	66
4.4.1	Discussion of Oscillations Model	66
4.4.2	Better Than Model...	68
4.5	Extension of Traffic Shaping Detection	68
4.5.1	Repeated Measurements	68
4.5.2	Outlook	68
4.6	Rate as a Function of Signal Strength and Time of Day	70
4.6.1	Reference Cell Measurements	70
4.6.2	Measurements in Live Network	73
4.7	Test Shortening	76
5	Crowdsourced Data for Network Performance Metrics	77
5.1	Structure of Open Data	77
5.1.1	Passively Active Measurements	77
5.1.2	Feature Filtering	77
5.1.3	Outlook: Possible Solutions	78
5.2	Tariff Limitation	78
5.2.1	Controlled Measurements	79
5.2.2	Open Data	79
5.3	Operator Benchmarking Using the RTR Data Set	81

5.3.1	Filtering Spurious Test Entries	82
5.3.2	Conclusion	82
Summary and Outlook		84
A	Volumes and Rates	86
A.1	Alternative Merging Algorithm	86
A.2	Comparison of Both Merging Algorithms	87
A.2.1	Data Rate Examples	88
A.3	Mean Squared Difference, Maximum Difference	90

Introduction

Motivation

In the recent years the telecommunication industry shifted their benchmarking portfolio from pure drive tests to an outsource of performance measurements to the end user. In this context we speak about crowdsourcing the benchmarking task.

In this approach the work is divided between several participants to achieve a result with combined efforts. The participants in that case are an undefined group of people out in the public. In the research community since the early 2010 many research groups started to adopt the methods to collect valuable information from end users, e.g. performance results, see [1].

In Austria the Austrian Regulatory Authority for Broadcasting and Telecommunications (RTR) is operating a system (RTR-NetTest) collecting data from end users. This data is provided as open data for download to any interested party. The data set contains not only performance results of each end user test, but also further meta information, e.g. the signal strength of the UE throughout the measurement.

RTR-NetTest is based on TCP over IP throughput measurements. Because our ultimate goal is analysis of RTR's open data, we focus on the same setup in the whole thesis. Another reason why to prefer TCP is UDP throttling observed in networks of some ISPs [2]. All controlled measurements performed in this work are static.

In this thesis we answer the research question, to which extent crowdsourced results of data rate measurements in LTE networks allow via postprocessing to derive appropriate benchmark metrics. In order to build new metrics on crowdsourced results, for which many factors are unknown (was the measurement static, was there any crosstraffic caused by other users or by the user him/herself, what was the cell load, was there any handover etc.), we perform our own measurements to calibrate and reference what we see in the open data.

Thesis Overview

In chapter 1 we give an overview of two important standards for throughput measurements and investigate some properties of our network under test (NUT). The first standard is by IETF and is focused on TCP throughput testing in managed business-class IP networks. The second standard is by 3GPP, it focuses on mobile networks, more specifically on UE average application layer data rates in a repeatable lab-based environment, and reuses some of IETF's ideas. Measurements performed in live network are more realistic but loosening standards' requirements leads to lower reproducibility:

- Lab environment: Most accurate, most reproducible.
- Drive tests: Performed by engineers in live networks, special equipment needed, mainly outdoor, high operational expenditure (OPEX) for operator [3].

- Measurements with conventional UEs: Our case. Minimization of drive tests (MDT) [4] has been released by 3GPP to support operator’s OPEX outsourcing. MDT standard specifies how UE logs measurements which it needs to collect anyway.

Since we are interested in measurements in live LTE network with conventional UEs, it is clear that we can’t fulfill all requirements given by the standards, still we need to fulfill as many as feasible. In crowdsourced measurements the situation is even more challenging because we often can’t verify which requirements were satisfied and which were violated. Now, being aware of the limitations and room for improvement we can proceed with more controlled measurements.

In chapter 2 we develop a systematic approach for processing data rate time series to be able to handle results of different tools in the same way. We have to master resampling, rebinning, meaningful representation of data rate as a time continuous trend function, merging data rate series of multiple connections in order to estimate the aggregate data rate. We detect tariff limits and discuss also some problems which occur later when using specific tools: removing data rate oscillations caused by unknown time offset of separate TCP connections, compression of data rate series, etc. Even though we later apply the concepts developed in this chapter to specific problems, we keep the whole chapter as general as possible.

In chapter 3 we introduce Android application called CMPT (Crowdsourcing Mobile Performance Tool) which we developed in order to schedule and execute our own measurements as well as measurement tools of third parties. CMPT also collects information about user’s mobile cell and neighboring cells (signal strength, cell id, etc.), about battery state, device memory, CPU load,... CMPT uploads all results to CouchDB database which is JSON-based in order to avoid fixed column structure and allow more flexibility during development. We also briefly mention different third-parties’ tools which we used: FLARP, iPerf3, Open-RMBT.

In chapter 4 we apply the concepts of chapter 2 to our controlled measurements. We analyze removing of oscillations caused by different time offsets of RTR-NetTest’s TCP connections and discuss the impact of RTR’s compression algorithm. We propose an extension to tariff limitation detection algorithm. And finally, analyze diurnal patterns in long-term repeated measurements.

In chapter 5 we make use of crowdsourced open data [5] collected by RTR-NetTest and made public by RTR. Because of very rough subsampling of the data rate series of available results we will have to modify algorithm for detection of tariff limits. Even though we managed to solve impairments of OpenData (chapter 3), future solutions need to be lightweight and robust to allow fast processing of millions of test results. We propose one example of such lightweight metric for comparison of different operators and we will recognize that automatized tests are present in networks of some operators.

Chapter 1

Benchmarking Methodology for Mobile Cellular Networks

1.1 Introduction

This chapter summarizes necessary minimum required for understanding of TCP throughput measurements in mobile LTE networks. In section 1.2, to avoid confusion and ambiguities, we list some key terms which are in different contexts used synonymously

Sec. 1.3 introduces LTE and physical layer parameters which are collected by CMPT (see chapter 3). In sec. 1.4 we will discuss TCP thoroughly.

Section 1.5 introduces two important standards – IETF’s standard for TCP throughput testing and 3GPP’s standard for UE application layer data throughput testing.

In sec. 1.6 we analyze, based on first measurements, TCP related effects.

Finally in sec. 1.7 we perform first automatized measurements. At this point it will become clear that it is not possible to fulfill all requirements of measurement standards for wired networks and lab based environment when measuring in live mobile network. We can however perform repeated measurements to analyze differences between, e.g. different numbers of TCP connections or between different applications.

1.2 Terminology

BS:	Base station. Since we focus on LTE networks only, we will use BS and eNodeB interchangeably.
RAN:	Radio access network. We will use it interchangeably with EUTRAN (evolved universal terrestrial radio access network).
UE:	User equipment. For most of the measurements in this thesis we used smart-phones LG F60 and LG K4.
DL and UL:	Downlink (/uplink) denotes the connection used for signal transmission from UE to BS (/ from BS to UE). We use the same abbreviations also for directions of data rate tests, i.e. DL for download (data flow from server to client) and UL for upload (from client to server).
Bandwidth, rate:	We will prefer the term data rate (or rate), expressed in bit/s, whenever possible, in order to avoid confusion with frequency bandwidth, expressed in Hz. However, we need to reference quantities bottleneck bandwidth (BB) and bandwidth delay product (BDP) presented in RFC 6349 [6] and therefore bandwidth and data rate will be used interchangeably in this chapter.

Peak data rate: Also bottleneck bandwidth (BB) [6], or maximum achievable (data) rate. The lowest data rate along the complete path between the sender and receiver.

Flight size, in-flight: The amount of data that has been sent but not yet cumulatively acknowledged: $\text{FlightSize} = \text{last bytes sent} - \text{last byte acknowledged}$. The term flight size (or FlightSize) is used in RFC 5681 [7]. The same quantity is called in-flight bytes, “commonly referred to as the send socket buffer” in RFC 793 [8] or also transmit window.

1.3 LTE

LTE (Long Term Evolution) is mobile network standard released by 3GPP (3rd Generation Partnership Project). It is based on OFDMA (orthogonal frequency division multiple access).¹ With higher order modulation (max. 64QAM) and large bandwidths (up to 20 MHz) high data rates can be achieved. The highest theoretical peak data rate on the transport channel is 75 Mbit/s in UL and 300 Mbit/s (in case of 4x4 spatial multiplexing) in DL [9]. QoS (quality of service) provisions allow transfer latency < 5 ms in the RAN.

1.3.1 Physical Layer Measurements at UE

Here we give brief overview of several physical layer related parameters which can be accessed in Android from application layer. Android’s class `CellSignalStrengthLte` allows us to access following parameters: RSRP, RSRQ, RSSNR, CQI and TA [10].

Unfortunately RSSNR, CQI and TA were reported just as 2147483647 (highest 32-bit signed value) on all devices we’ve ever tested (LG F60, LG K4, several Samsung devices, Ulefone Power,...).

RSRP (Reference Signal Received Power)

Average over the power contributions (in [W]) of the resource elements that carry cell-specific reference signals within the considered measurement frequency bandwidth [11].

I.e. $\text{RSRP} = \frac{1}{N} \sum_{i=1}^N P_{\text{RS},i}$ where $P_{\text{RS},i}$ is the power of i -th resource element carrying reference signal and N is the total number of REs which contain RS. RSRP is calculated only over those REs which contain RS. It provides information about DL path loss.

RSSI (Received Signal Strength Indicator)

E-UTRA Carrier Received Signal Strength Indicator (RSSI), comprises the linear average of the total received power (in [W]) observed only in OFDM symbols containing reference symbols for antenna port 0, in the measurement bandwidth, over N number of resource blocks by the UE from all sources, including co-channel serving and non-serving cells, adjacent channel interference, thermal noise etc [11].

RSRQ (Reference Signal Received Quality)

Ratio $N \cdot \text{RSRP} / \text{RSSI}$, where N is the number of RBs of the E-UTRA carrier RSSI measurement bandwidth. The measurements in the numerator and denominator shall be made over the same set of resource blocks [11].

¹In DL. In UL SC-FDMA (single carrier frequency division multiple access) is used.

CQI (Channel Quality Indicator)

CQI is a four-bit number reported by UE to the BS to indicate the channel quality. Based on CQI the BS picks modulation and coding scheme (MCS) which determine the DL data rate [12].

SINR (Signal to Interference and Noise Ration)

According to [13]: SINR is not standardized by 3GPP (therefore it is not reported to the BS) but is measured by UEs (definition is vendor specific, SINR may be measured over different REs) to internally determine the CQI because it better quantifies the relationship between RF conditions and Throughput

The Android's class provides method `getRssnr`, the documentation is unfortunately not very detailed: "`getRssnr()`: Get reference signal signal-to-noise ratio," so we can't be sure whether this is SINR value on which the CQI calculation is based.

TA (Timing Advance)

Timing advance is reported by BS to the UE to adjust UL signal transmission timing.

1.4 TCP

Transmission Control Protocol (TCP) is a transport protocol, which provides a reliable, connection-oriented service to the invoking application. One end of the TCP connection is attached to the client socket, the other end is attached to a server socket. TCP socket is identified by a four-tuple (source IP address, source port number, destination IP address, destination port number).

This section, mostly based on [8], [7], [6] and [14], summarizes some of TCP's concepts which will be referenced in later sections. The fundamentals, like header format, three way handshake, TCP reliability, closing a connection, etc. are assumed to be well known and they will not be presented here.

1.4.1 Round-Trip Time

Round-trip time (RTT) is the elapsed time between the clocking in of the first bit of a TCP segment sent and the receipt of the last bit of the corresponding TCP ACK.

The RTT dataset needs to be baselined during off-peak hours in order to obtain a reliable figure of the inherent network latency. Otherwise, additional delay caused by network buffering can occur. The minimum measured value serves as the baseline round-trip time RTT_{baseline} . This will most closely estimate the real inherent RTT [6].

The most accurate method for estimating RTT would be using test equipment on each end of the network, so that a packet stream can be measured from end to end.

Easier to realize but less accurate are ICMP pings. Some limitations with ICMP ping may include ms resolution and whether or not the network elements are responding to pings. Also, ICMP is often rate-limited or segregated into different buffer queues. ICMP is not as reliable and accurate as in-band measurements.

1.4.2 Bandwidth-Delay Product

Bandwidth-delay product (BDP) is the product of a data link's capacity (BB) and its end-to-end delay (inherent RTT of non-congested network):

$$BDP = BB \cdot RTT_{\text{baseline}}.$$

BDP is calculated to provide estimates of the TCP RWND (see next subsection) and send socket buffer sizes that are used in data rate tests. BB can be achieved only if TCP connection is able to fill network's BDP.

1.4.3 Flow Control

Flow control means matching of data rate at which sender is sending against the rate at which the receiver application is reading. The maximum number of unacknowledged bytes the sender can transmit is given by a variable called the receive window (RWND):

$$\text{FlightSize} \leq \text{RWND}. \quad (1.1)$$

RWND is fed back from receiver to sender in ACK messages and corresponds to the number of free bytes in receiver's receive buffer.

Window Scaling

The RWND field in the TCP header is two bytes [8], the maximum RWND value is thus

$$\text{RWND}_{\max} = (2^{16} - 1)\text{B} = 65535\text{B} = 64\text{KiB} - 1\text{B}.$$

As pointed out in [15], for a standard 64 KiB RWND and 20 ms RTT, the achievable data rate would be limited at ca $\frac{65535\text{B}}{20\text{ms}} = 26.214\text{Mbit/s}$ for a single TCP connection. This limitation can be mitigated with multiple concurrent TCP connections or with enlarging the RWND through TCP window scaling [16].

As specified in [8]: In the options field of TCP header, the option-kind is one byte, the option-length is one byte and the option-data is (option-length - 2) bytes. I.e. option-length determines the length of whole option, including option-kind and option-length fields.

RFC 7323 [16] introduces window scale option: option-kind = 3, option-length = 3, option-data = WindShift. If this option is used, the value $\text{RWND}_{\text{scaled}}$, calculated by left-shifting the RWND by WindShift bits,

$$\text{RWND}_{\text{scaled}} = \text{RWND} \cdot 2^{\text{WindShift}},$$

is used for flow control. The scale option may be sent in a SYN segment and SYN-ACK segment. Sender's WindShift and receiver's WindShift values remain unchanged during the whole connection.

WindShift = 0 means no scaling. The maximum allowed value is WindShift = 14. It follows that maximum receive window size is

$$\text{RWND}_{\text{scaled},\max} = (2^{14+16} - 1)\text{B} = 1\text{GiB} - 1\text{B}.$$

To make the notation more concise, also the scaled receive window (if window scaling enabled) will be denoted as RWND.

1.4.4 Congestion Control

Congestion control means that each sender limits the rate at which it sends traffic into its connection based on perceived network congestion. The sender keeps track of an additional variable – congestion window (CWND). The amount of unacknowledged data at the sender is limited as follows:

$$\text{FlightSize} \leq \min\{\text{CWND}, \text{RWND}\}.$$

The RFC 5681 [7] specifies four congestion control algorithms: slow start, congestion avoidance, fast retransmit and fast recovery.

Slow Start Threshold

Another state variable, the slow start threshold (sssthresh), determines which algorithm is used to control data transmission. Slow start algorithm is used when $CWND < sssthresh$, congestion avoidance algorithm is used when $sssthresh > CWND$.² The initial sssthresh value can be set arbitrarily high but it must be reduced in response to congestion: if sender detects segment loss (retransmission timer timeout), the value of sssthresh must be set to no more than

$$sssthresh = \max\{\text{FlightSize}/2, 2 \cdot \text{MSS}\}, \quad (1.2)$$

where Maximum Segment Size (MSS) is the size of the largest segment that the sender can transmit (it can be based on MTU of the network, the path MTU discovery algorithm – RFC 4821 [17] – or other factors) and FlightSize is the amount of data that has been sent but not yet cumulatively acknowledged.

Slow Start

The slow start algorithm is used at the beginning of a transmission into a network with unknown conditions or after repairing loss detected by the retransmission timer in order to slowly probe the network to determine the available capacity and avoid congesting the network with an inappropriately large burst of data. At the beginning of a transmission $CWND$ is set to IW (Initial Window). Upon receipt of an ACK covering new data, the recommended (RFC 5681 [7]) increase of $CWND$ is:

$$CWND += \min\{N, \text{MSS}\},$$

where N is the number of previously unacknowledged bytes acknowledged in the incoming ACK. Details regarding the size of IW are specified in RFC 3390 [18]

Congestion Avoidance

Congestion avoidance algorithm increases the size of $CWND$ less rapidly, ca one full-sized segment per RTT. During congestion avoidance, $CWND$ must not be increased by more than MSS bytes per RTT.

The recommended way is to count the number of bytes that have been acknowledged by ACKs for new data (an additional state variable has to be maintained) – when the number of bytes acknowledged reaches $CWND$, then $CWND$ can be incremented by up to MSS bytes.

Upon a timeout $CWND$ must be set to no more than the loss window, LW , which equals one full-sized segment (regardless of the value of IW). Therefore, after retransmitting the dropped segment the TCP sender uses the slow start algorithm to increase the $CWND$ from LW to the new value of $sssthresh$, at which point congestion avoidance again takes over.

Fast Retransmit

The fast retransmit algorithm uses the arrival of three duplicate ACKs as an indication that a segment has been lost. After receiving three duplicate ACKs, TCP performs a retransmission of what appears to be the missing segment, without waiting for the expiration of that segment's retransmission timer (the timeout period can be relatively long).

²When $sssthresh = CWND$, sender may use either slow start or congestion avoidance.

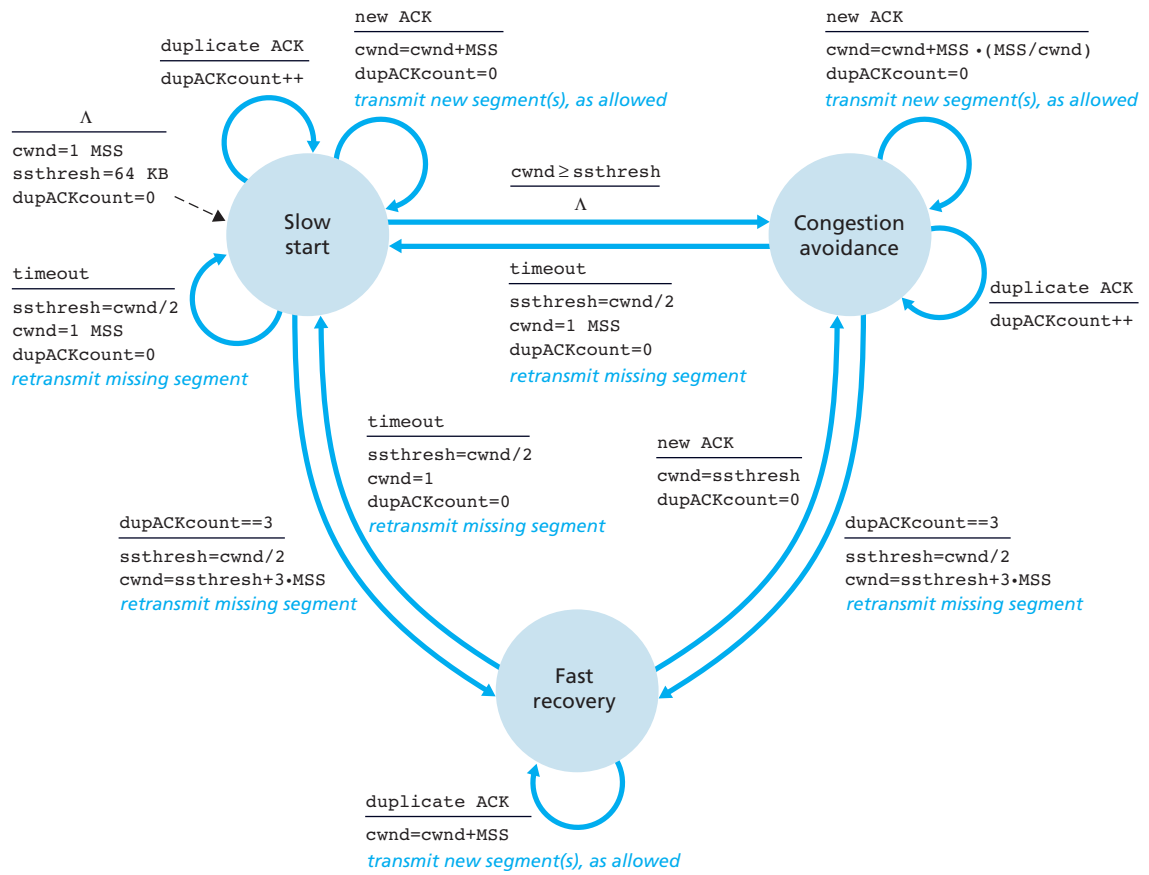


Figure 1.1: State diagram of TCP congestion control algorithm. Source: [14], figure 3.52.

Fast Recovery

The receiver can generate a duplicate ACK only when a segment has arrived, i.e. that segment has left the network and is in the receiver's buffer, it is no longer consuming network resources.³ This is the reason why, after the fast retransmit, the fast recovery algorithm (instead of slow start) governs the transmission of new data until a non-duplicate ACK arrives.

After receiving the third duplicate ACK, $ssthresh$ is set to no more than allowed by eq. (1.2). The “lost” segment is retransmitted and $CWND$ is set to $ssthresh + 3 \cdot MSS$. This artificially inflates the $CWND$ by the number of segments that have left the network. For each additional duplicate ACK received (after the third), $CWND$ is incremented by MSS ($CWND$ again inflated to reflect the additional segment leaving the network).

When previously unsent data is available and the new value of $CWND$ and the $RWND$ allow, TCP sends $1 \cdot MSS$ bytes of previously unsent data.

When the next ACK acknowledging previously unacknowledged data arrives, $CWND$ is set to $ssthresh$. This is termed “deflating” the window. This ACK serves as the acknowledgment elicited by the retransmission of the “lost” segment – one RTT after the retransmission. Additionally, this ACK should acknowledge all the intermediate segments sent between the lost segment and the receipt of the third duplicate ACK, if none of these were lost.

Fast recovery is a recommended, but not required, component of TCP. An early version, TCP Tahoe, enters the slow start phase after triple duplicate ACK. The newer version, TCP Reno, incorporated fast recovery.

1.4.5 Overview of Congestion Avoidance Algorithms

This subsection is not an exhaustive list of all congestion avoidance algorithms. We summarize TCP Tahoe and Reno, which were already mentioned in previous section. We mention also New Reno, which was used by default in Linux kernels before 2.6.8 and SACK which allows retransmission of multiple lost segments in one RTT and is widely deployed. BIC is used by default in Linux kernels 2.6.8 – 2.6.18. CUBIC is used by default in Linux kernel 2.6.19 and later.

This subsection is based on [14], [19], [20], [21], [22] and [23]. List of many other algorithms can be found for example at [24].

TCP Tahoe

- 3 duplicate ACKs \Rightarrow fast retransmit, $ssthresh = \frac{CWND}{2}$, $CWND = 1MSS$, slow start
- ACK timeout (retransmission timeout, RTO) \Rightarrow slow start, $CWND = 1MSS$

TCP Reno

Adds fast recovery in order to not empty $CWND$ every time a packet is lost:

- Three duplicate ACKs \Rightarrow fast retransmit, $CWND = \frac{CWND}{2}$, $ssthresh = CWND$, fast recovery (wait for ACK of entire transmit window, then congestion avoidance; or timeout, then slow start)
- RTO \Rightarrow slow start, $CWND = 1MSS$

³Although a segment duplication by the network can invalidate this conclusion.

TCP New Reno

Extension of TCP Reno, improved retransmission during fast recovery phase:

- if entire transmit window acknowledged \Rightarrow $CWND = ssthresh$, congestion avoidance
- if partial ACK \Rightarrow assume that the next segment was lost, retransmit that segment, set number of duplicate ACKs to zero, reset timeout timer

Compared to Reno, the New Reno is able to fill multiple gaps in the transmit window, i.e. it overcomes the problem of reducing $CWND$ multiple times.

TCP SACK

Selective ACKs instead of cumulative ACKs: Each ACK has a block which describes which segments are being acknowledged \Rightarrow multiple lost segments can be sent in one RTT. Disadvantage: SACK requires modification of the receiver.

TCP BIC

Binary Increase Congestion control. It solves the problem of slow response (i.e. unused bandwidth) of TCP in fast long distance networks.

Main feature of BIC is unique window growth function. The algorithm tries to find the maximum where to keep the window at for a long period of time, by using a binary search algorithm [25].

TCP CUBIC

The window growth function of BIC, which consists of different phases (additive increase, binary search), is replaced by cubic function

$$CWND_{cubic} = C(t - K)^3 + CWND_{max},$$

where $CWND_{max}$ is window size just before the last window reduction, K and C are some constants and t is the elapsed time from the last window reduction \Rightarrow $CWND$ growth is thus independent of RTT.

CUBIC retains stability and scalability of BIC and simplifies the window control.

1.5 Active Performance Measurements in Standardization

RFC 6349 [6] focuses on TCP throughput testing in managed business-class IP networks. TR 37.901 [26] introduces procedures for measuring average application-layer data rate under simulated realistic network scheduling and radio conditions in a repeatable lab-based environment

1.5.1 IETF – Framework for TCP Throughput Testing

TCP testing is performed in addition to traditional layer 2/3 tests such as RFC 2544 [27] (updated by RFC 6201 [28] and RFC 6815 [29]) or other methods of network stress tests which are required to verify the integrity of the network before conducting TCP tests. Examples of layer 2/3 tests: iPerf (UDP mode) and manual packet-layer test techniques where packet throughput, loss, and delay measurements are conducted.

It is not possible to make an accurate TCP Throughput measurement when the network is dysfunctional – in case of high packet loss and/or high jitter the TCP throughput testing

will not be meaningful. As a guideline, 5% packet loss and/or 150 ms of jitter may be considered too high for an accurate measurement.

The framework is designed for measuring end-to-end TCP throughput in managed business-class IP networks, e.g. Ethernet-terminated services with Service Level Agreement (SLA) provided from the network operator – TCP throughput must achieve the data rate guaranteed by the SLA.

End-users with “best effort” access could use this methodology, but this framework and its metrics are intended to be used in a predictable managed IP network.

TCP Throughput

The achievable TCP throughput is that amount of data per unit of time that TCP transports when in the TCP equilibrium state. It can be estimated from RWND and RTT:

$$\text{Throughput} = \frac{\text{RWND}}{\text{RTT}},$$

assuming that RWND size is large enough (TCP connection is able to fill network’s BDP) – if a smaller RWND is used, then the TCP throughput cannot be optimal.

The TCP throughput test is required to characterize performance at different times of the day and also to last long enough to properly exercise network buffers (greater than 30 seconds).

Methodology

1. Identify the path maximum transmission unit (MTU) so that the test device is configured properly to avoid fragmentation during all subsequent tests. A robust method for path MTU discovery is standardized in RFC 4821 [17].
2. Determine the $\text{RTT}_{\text{baseline}}$ and BB, calculate the BDP, estimate the RWND and send socket buffer sizes that is used.
3. TCP connection throughput tests.

Metrics

Definition 1.5.1. Transfer time ratio (TTR): The ratio between the actual TCP transfer time (ATT) versus the ideal TCP transfer time (ITT). ATT is the time it takes to transfer a block of data across TCP connection(s). ITT is the predicted time for which a block of data should transfer across TCP connection(s), considering the BB.

$$\text{TTR} = \frac{\text{ATT}}{\text{ITT}}$$

The ITT is derived from the maximum achievable TCP throughput, which is related to the BB and layer 1/2/3/4 overheads associated with the network path.

Definition 1.5.2. TCP efficiency η : Percentage of bytes that were not retransmitted.

$$\eta = \frac{\text{transmitted bytes} - \text{retransmitted bytes}}{\text{transmitted bytes}}$$

Transmitted bytes are the total number of TCP bytes to be transmitted, including the original and the retransmitted bytes.

Network congestion causing packet loss may be inferred from a poor TCP Efficiency. Higher η means less packet loss.

Definition 1.5.3. Buffer Delay Percentage: Represents the increase in RTT during a TCP throughput test versus the inherent (baseline) RTT.

$$\text{buffer delay percentage} = \frac{\overline{\text{RTT}} - \text{RTT}_{\text{baseline}}}{\text{RTT}_{\text{baseline}}}$$

The average round-trip time $\overline{\text{RTT}}$ is derived from the total of all measured RTTs during the whole test and from the test duration T .

$$\overline{\text{RTT}} = \frac{\sum_i \text{RTT}_i}{T}.$$

Network congestion causing an increase in RTT may be inferred from the buffer delay percentage (0% means no increase in RTT over baseline).

1.5.2 3GPP – UE Application Layer Data Throughput Performance

3GPP TR 37.901 [26] defines test procedures to measure UE average application layer data rates in a repeatable lab-based environment using lab-based simulators (faders, AWGN sources, etc.).

Definition 1.5.4. The measured UE application layer throughput T is the number of useful user data bits per unit of time delivered by the network from the source end point to the destination end point, excluding protocol overhead (TCP, UDP, etc.) and retransmitted data packets.

- Radio connection is limited to LTE and W-CDMA Rel-5 (HSDPA).
- For TCP tests only the FTP application protocol is proposed in order to reduce the amount of testing (e.g. HTTP testing is seen as redundant, similar results are expected).
- For UDP the raw data transfer is proposed (no streaming protocol) to simplify the UDP transfer application requirements.
- Tests are performed separately in DL, UL and bi-directional.

Test Times

In UTRAN Each minimum test time is derived from the speed in the profile. The longest time 164 s for the slow speed fading profiles. In LTE the minimum test time is simulated for each test case.⁴ Most of the layer 1 receiver and performance tests in UTRAN and LTE are governed by the test time due to fading.

Since the statistics of T are unknown it is not possible to give a variance of the measured throughput around the true throughput, i.e. it is not possible to give a confidence level of the measurement when given a predefined test time and to calculate a minimum test time for a given confidence level.

Therefore, for HSPA, the recommendation for minimum test time is 164 seconds which is derived from lowest speed fading (crossing of 990 wavelengths when travelling with the speed given in the fading profile). For static testing, 60 seconds is recommended.

For LTE, the recommended minimum test time for static testing is also set to 60 s.

⁴Longest: 150 000 minimum number of active samples = 1500 s net test time.
Shortest: 1366 minimum number of active samples = 13.66 s net test time.

Limitations

Here we briefly present requirements recommended in the standard which are however out of our control when measuring in live LTE network. For different scenarios the standard defines specific:

- signal levels,
- fading profiles (for LTE: static, EPA5, EVA5,...),
- noise and interference levels,
- received signal energy per resource element (RE) at antenna port,
- power spectral density of a white noise at antenna port,
- downlink power allocation,
-

1.6 First Analysis: TCP Related Effects

In this section we will discuss the first measurements and analyze how many parallel TCP connections are required to properly measure the achievable data rate in the network under test (NUT).

We will see that full throughput is already achieved with a single connection due to TCP window scaling, we will encounter the consequences of large buffers and finally we learn that different number of TCP connections changes initial ramp up phase because of congestion control algorithm.

1.6.1 Measurement Setup

To get more information about the NUT we performed several active TCP measurements with iPerf3 [30] on an Android smart phone LG F60. For capturing packets from cellular connection we used *tcpdump* binary for Android [31]. Packet captures were analyzed with *Wireshark* [32]. With Android application *Kernel Adiutor* [33] we found out that UE uses TCP CUBIC algorithm. This is no surprise since Android is based on Linux kernel and CUBIC is the default TCP algorithm in Linux kernel since version 2.6.19, as mentioned in subsection 1.4.5.

We measured in live LTE network of operator A1, using data rate unlimited tariff. The expected bottleneck is in radio access network (RAN), since wired connections in the backbone network to which both, operator and university network, are connected, are usually overprovisioned.⁵ We used our own iPerf3 server connected to university network in order to make sure, there are no parallel tests running on the server. We use Linux server, TCP algorithm is also CUBIC.

1.6.2 Flow Control Limitation

Each TCP connection has its own RWND. The bottleneck data rate can be achieved only if TCP connections are able to fill NUT's BDP, i.e. if

$$\sum_{i=1}^N \text{RWND}_i \geq \text{BDP},$$

⁵This will be confirmed by measurements in reference cell which will be discussed later.

where $RWND_i$ denotes the RWND of i -th TCP connection and N the total number of TCP connections.

The minimum RTT which we measured during off-peak hours (7:20–7:25, Tuesday: March 7, 2017) was $RTT_{\text{baseline}} = 18$ ms. We can thus conclude that the real inherent network RTT is not larger than 18 ms.

The UE we use is LTE UE category 4 [34], i.e. the maximum data rate it is capable of is $R_{\text{DL,max}} = 150$ Mbit/s in DL and $R_{\text{UL,max}} = 50$ Mbit/s in UL [35].⁶ We do not expect higher data rates in our NUT,⁷ whose bottleneck is in the radio access network (RAN). Using these values, we can calculate upper bounds for BDP in both directions:

$$BDP_{\text{DL}} \leq R_{\text{DL,max}} \cdot RTT_{\text{baseline}} \leq \frac{150 \cdot 18}{8} \text{ kB} = 337.5 \text{ kB}, \quad (1.3)$$

$$BDP_{\text{UL}} \leq R_{\text{UL,max}} \cdot RTT_{\text{baseline}} \leq \frac{50 \cdot 18}{8} \text{ kB} = 112.5 \text{ kB}. \quad (1.4)$$

In our setup, both – the server and the UE – support TCP window scaling. In DL the UE’s RWND of a single TCP connection reaches 522.88 kB in less than 0.2 s (fig. 1.3) and in UL the server’s RWND of a single TCP connection reaches 185.856 kB in less than 0.2 s (fig. 1.2). In both directions RWND further increases – in DL up to 1019.392 kB, in UL up to 996.224 kB.

We can conclude that RWND (i.e. flow control, eq. (1.1)) is not the limiting factor in our data rate measurement, after 0.2 ms the RWND is larger than BDP in both directions. From the perspective of flow control there is no need to use more than one connection.

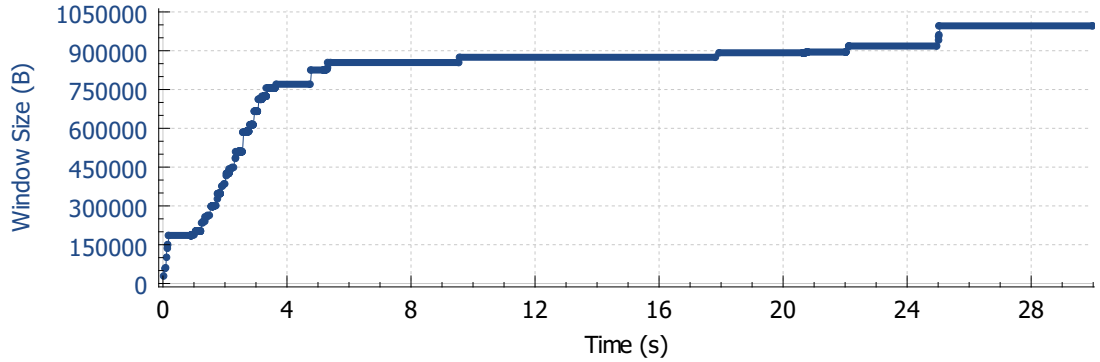


Figure 1.2: Wireshark trace of server’s RWND during an iPerf3 UL test.

1.6.3 Impact of Bufferbloat

Examining buffers was not within the primary scope of this work, however, we encountered it when measuring RTT_{baseline} . In this subsection we bring few brief comments.

In fig. 1.4 we can notice that RTT is increasing during the first three seconds and remains significantly higher than RTT_{baseline} for the rest of the test. This behavior is known as bufferbloat [36]: When buffers along the path start to fill, TCP overestimates available bandwidth because BDP appears to be larger due to additional queuing delays, i.e. CWND further increases although the throughput does not increase anymore, since BB was already reached earlier. This leads to increase of RTT without further increasing throughput.

⁶Maximum number of DL-SCH transport block bits received within a TTI is 150751. Maximum number of UL-SCH transport block bits transmitted within a TTI is 51024. SCH = shared channel. TTI = transmission time interval. TTI = 1 ms.

⁷If we would ever observed values which are close to this limit, we would have to use UE of different category to assure that UE is not the limiting factor in the data rate measurement.

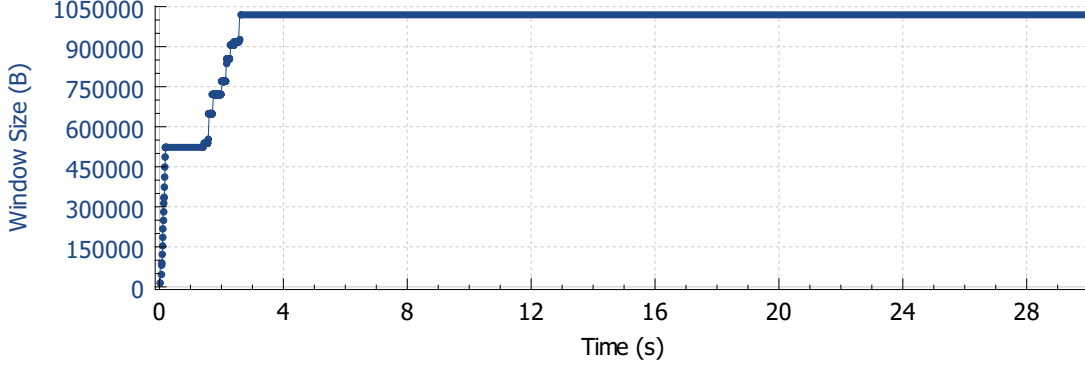


Figure 1.3: Wireshark trace of UE's RWND during an iPerf3 DL test.

In our measurement the situation is not so critical as in [36], where excessive buffers were present and RTTs larger than one second were measured. In fig. 1.4 the longest RTT is just 210 ms, the principle is however the same.

At the beginning of this test the RTT was ca 25 ms, after reaching its maximum (full buffer) it fluctuated around the mean of ≈ 120 ms. The corresponding throughput (fig. 1.5) was ≈ 20 Mbit/s. For both we consider the steady phase $t \in [16, 28]$ s. Denoting the RTT-increase caused by bufferbloat as

$$\Delta \text{RTT} \triangleq \overline{\text{RTT}} - \text{RTT}_{\text{baseline}} \approx 100 \text{ ms},$$

we can roughly estimate the sum of sizes of all buffers along the path as follows:

$$V = \Delta \text{RTT} \cdot \text{Throughput} \approx 2 \text{ Mbit} = 250 \text{ kB}.$$

If a packet arrives to a full buffer of size V , it has to wait for $\Delta \text{RTT} = \frac{V}{\text{Throughput}}$ until all previous packets will be departed. We can also calculate the buffer delay percentage $\approx 567\%$ according to def. 1.5.3.

The calculation above is based on an assumption that buffers along the path have constant size in terms of data volume. Interesting findings are presented in paper [37], where authors discovered that some mobile Internet service providers (ISPs) employ buffers which are not constant in terms of data volume but in terms of number of packets. This can be detected by several measurements with different packet sizes: For every packet size we would measure different data-volume-size of buffer. Dividing this data-volume-size by packet size would lead to the same number of packets in the buffer for every measurement, independently of chosen packet size.

For another ISP the authors discovered active queue management (AQM) – measured buffer size was linear function of throughput, packets that spent more than 800 ms in the buffer would be dropped.

Note that the cited paper used UDP only and it focused on buffers in DL, the same hypothesis may be applied in case of TCP. In our measurement we investigated UL, because it was easier to collect Wireshark traces at the UE and directly determine RTT based on the time difference between sent TCP fragment and corresponding ACK.

1.6.4 Congestion Control Limitation

Because the CWND is an internal variable which is not contained in TCP header and the packets were captured at the UE, we have to analyze UL test in order to see the CWND size. In this subsection we compare an iPerf3 UL test with one TCP connection to an iPerf3 UL test with five parallel TCP connections, we will call them the “1-connection test” and the “5-connection test.”

Fig. 1.10 shows the number of in-flight bytes of the 1-connection test, fig. 1.11 shows the number of in-flight bytes of a single connection of the 5-connection test. The green line, which is higher than in-flight bytes during the whole test, indicates the size of RWND – the number of in-flight bytes thus corresponds to CWND only. The RWND is not a limiting factor as already mentioned in the subsection “1.6.2 Flow Control Limitation.” Note: the test duration was 30 s in both cases, fig. 1.10, 1.11 show just first ≈ 12 s in order to see the initial phase in more detail, in the rest of the test the number of in-flight bytes looks very similar.

In fig. 1.6, 1.7 and 1.8 we compare TCP throughput of 1-connection and 5-connection test in different phases. We aligned both tests so that the first packet corresponds to $t = 0$. Fig. 1.6 shows details of both tests in different phases, we can recognize individual packet bursts.⁸ Fig. 1.7 displays smoothed tests – moving average (MA) with window size of 1 s – in order to visualize the throughput trend.

In fig. 1.8 attempted to characterize the initial ramp-up phase of both tests by a continuous function: we fitted linear and quadratic functions (denoted as “reference”) to the ramp-up phase of smoothed throughput (right plot) and then performed deconvolution (left plot). This representation has certain advantages: In fig. 1.6 it is not possible to easily recognize any throughput trend, whereas in fig. 1.7 it looks like that 5-connection test ramps-up continuously which is however misleading, because this successive increase is caused solely by the smoothing. After deconvolution of the reference we see a sudden jump in throughput of 5-connection test.

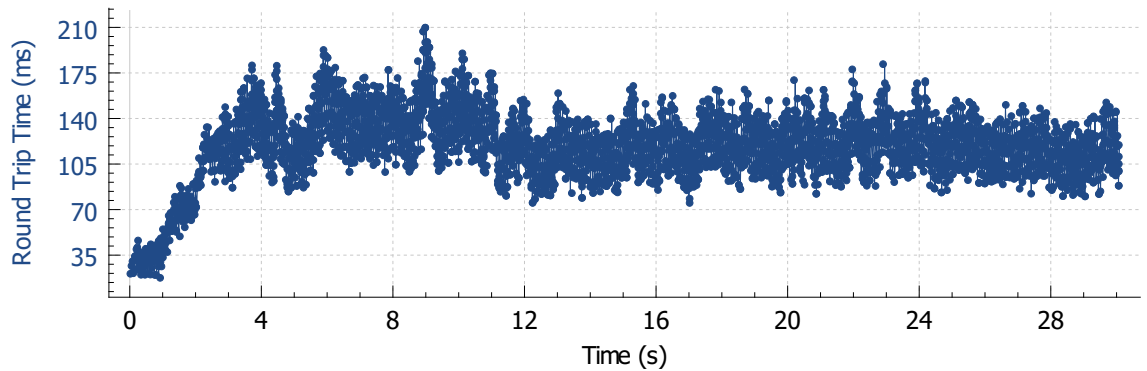


Figure 1.4: Wireshark trace of TCP RTT during an iPerf3 UL test.

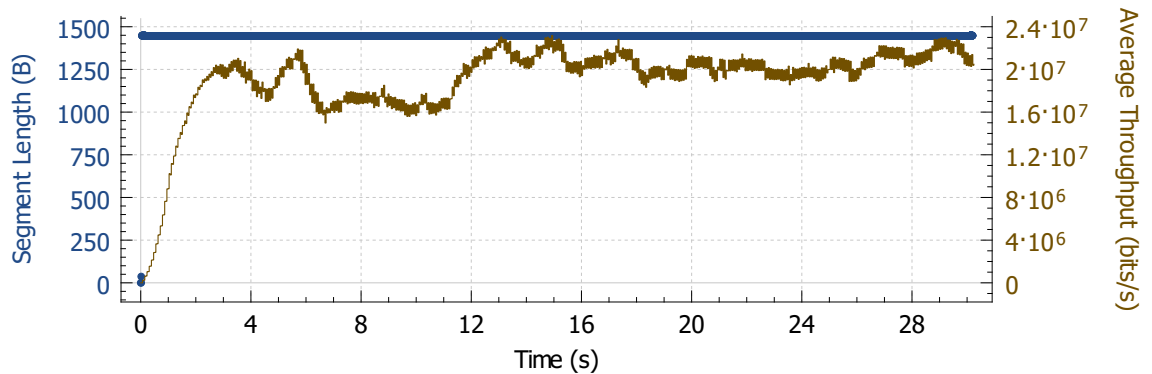


Figure 1.5: Wireshark trace of TCP throughput (right vertical axis) of the same test as in fig. 1.4. (Moving average with window size of 1 s.)

⁸Throughput traces were exported from Wireshark with the highest possible resolution: 1 ms. In fig. 1.6 we applied moving average with window size of 10 ms in order to broaden the bursts for easier visualization.

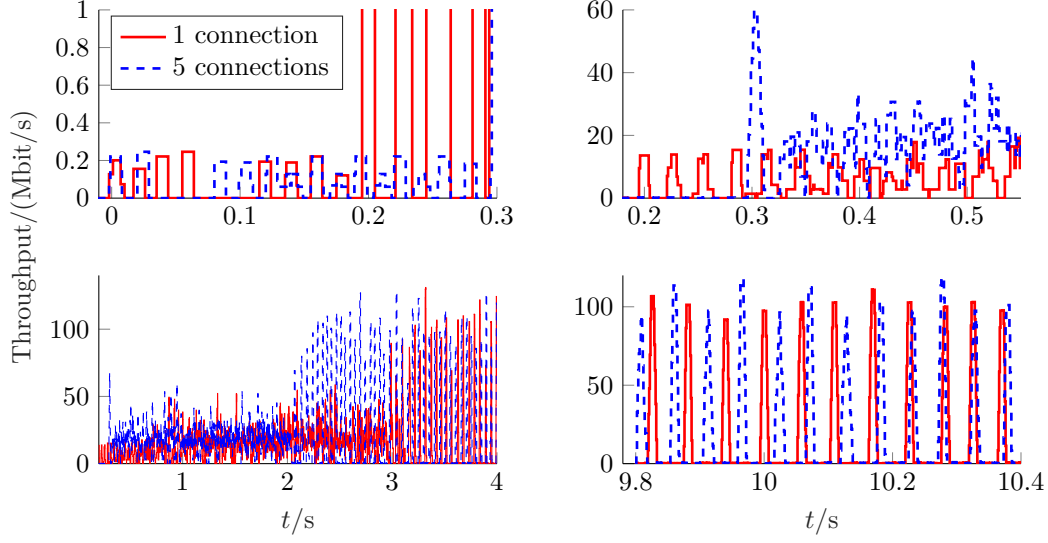


Figure 1.6: Comparison of TCP throughputs of iPerf3 UL test with a single TCP connection (red solid) and with five parallel connections (blue dashed). Each subplot is zoomed to different test phase.

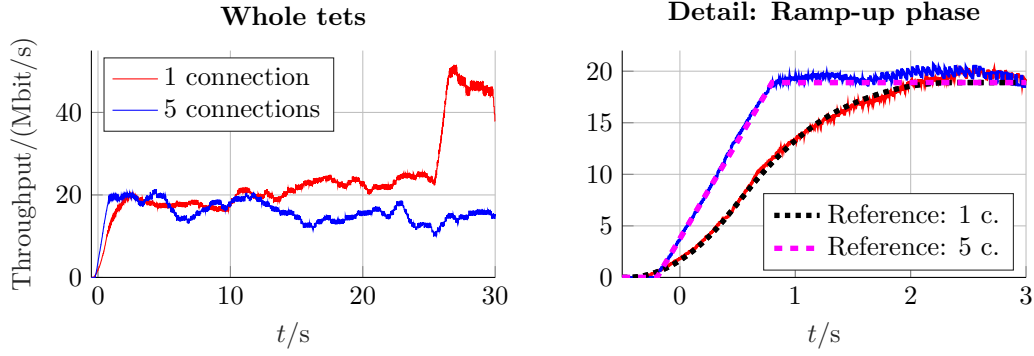


Figure 1.7: The same 1- and 5-connection tests as in fig. 1.6, this time smoothed using moving average with windows size of 1 s in order to visualize the throughput trend. Left: whole test; duration 30 s. Right: detail showing the initial ramp-up phase, including reference (see fig. 1.8).

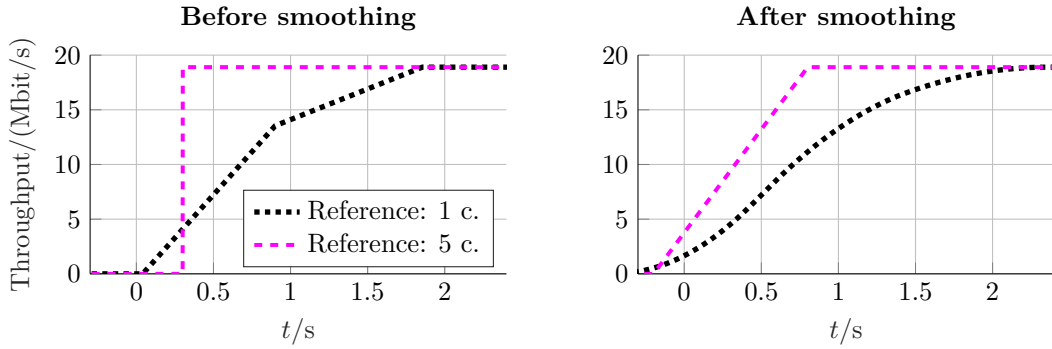


Figure 1.8: We fit linear and quadratic functions (right plot) to the smoothed throughput (fig. 1.7, right) in order to represent the ramp-up phase in terms of continuous functions. The left plot shows, how these fitted functions look like without smoothing.

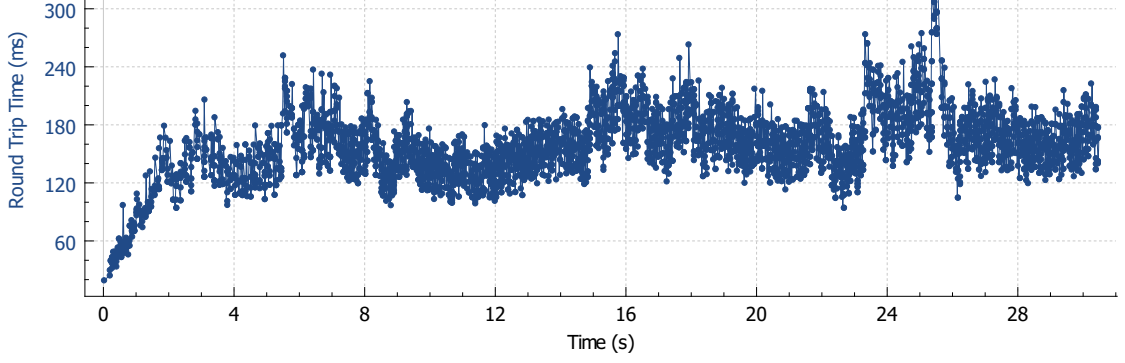


Figure 1.9: RTT of a single TCP connection during 5-connection iPerf3 UL test.

Detailed Analysis

The smoothed rates in fig. 1.7 are shown in order to see the long term trends and emphasize that throughput varies during whole test. The sudden increase in throughput of 1-connection test at $t \approx 25$ s might be caused by getting more resources at base station (BS), e.g. if other users release their resources or leave the cell.

Top left plot in fig. 1.6 shows first 300 ms of both tests. We first observe few smaller bursts. At $t = 0.2$ s the burst size of the 1-connection test significantly increases, in the 5-connection test this happens ≈ 100 ms later. In top right plot we see this 100 ms difference as well as the fact that the bursts of the 5-connection test are higher. The reference (fig. 1.8, left) shows that the 5-connection test reaches the full throughput almost immediately at $t = 0.3$ ms, whereas in case of the 1-connection test we observe approximately linear increase of the throughput – full throughput is reached at $t \approx 1.8$ s.

Another interesting thing happens at $t \approx 2$ s (5-connection test) and $t \approx 3$ s (1-connection test): bursts increase to more than double size (fig. 1.6, bottom left plot) although the throughput doesn't change much (fig. 1.7). Full throughput was already reached earlier. This is caused by bufferbloat: In fig. 1.4 and 1.10 after ≈ 3 s (1-connection test) and in fig. 1.9 and 1.11 after ≈ 2 s (5-connection test) the buffer is filled, i.e. RTT and CWND reach the maximum, don't increase anymore and fluctuate around quite stable average. Filling the buffers along the path results in higher RTT and larger CWND, because TCP overestimates the bottleneck bandwidth as explained in subsection 1.6.3.

Comparing fig. 1.4 (RTT of 1-conn. test) and fig. 1.9 (RTT of a single conn. of 5-conn. test), we can see that 5-conn. test has higher RTT. Comparing CWND in fig. 1.10 and fig. 1.11, we find out that 5-conn. test has lower CWND of a single connection. This is quite obvious, since the five connections in the 5-conn. test share the same path and bottleneck bandwidth, therefore the RTT of each separate connection has to be larger and throughput (CWND) has to be smaller than in case of the 1-conn. test. Note that CWND of the 1-connection test fluctuates around ≈ 300 kB, whereas a single connection of the 5-connection test around ≈ 60 kB, which is one fifth – this was expected since TCP assures fair share among multiple connections.

If we compare behavior of 1-conn. and 5-conn. test around $t = 10$ s (fig. 1.6, bottom right) where both tests have the same throughput (fig. 1.7, left), we see no difference – both tests show bursts of approximately same size and spacing. Different number of connections has no visible impact in this phase.

Conclusion

It is not wise to draw general conclusions from single tests. It is however good to validate what to expect before we continue with systematic testing and processing of hundreds or

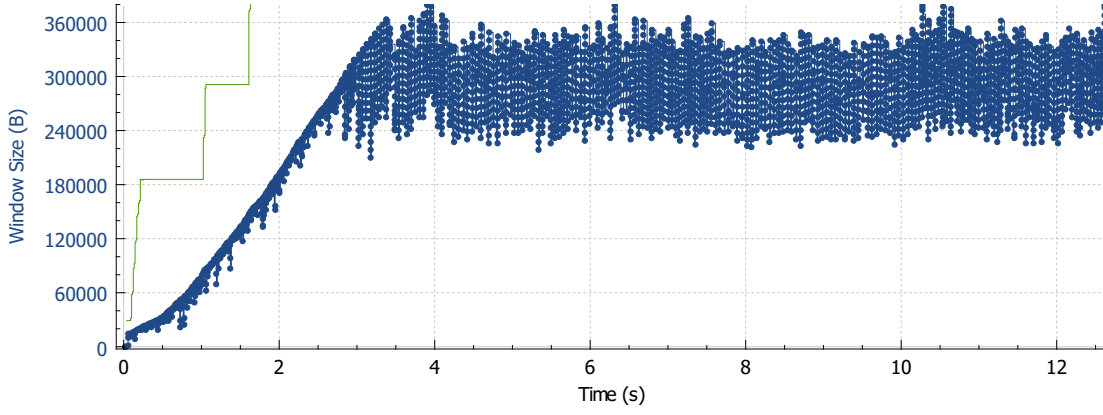


Figure 1.10: CWND size during the 1-connection iPerf3 UL test. Green line = RWND size.

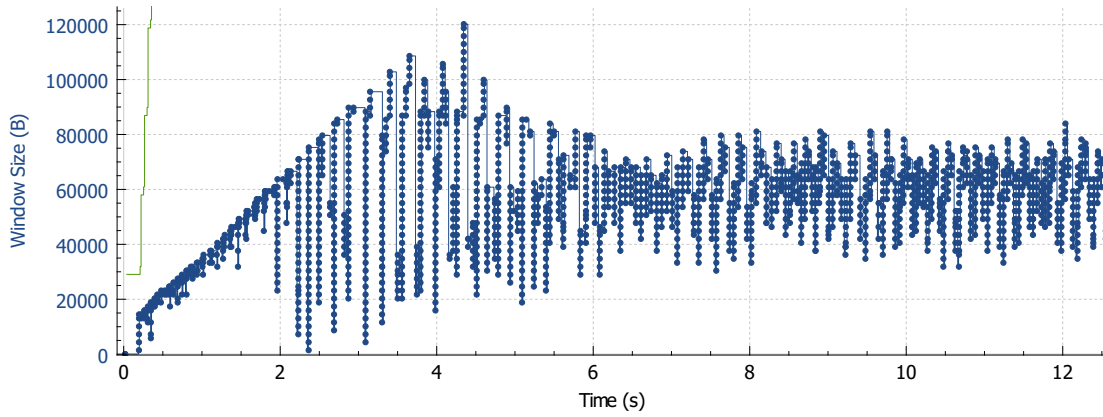


Figure 1.11: CWND size of a single TCP connection during the 5-connection iPerf3 UL test. Green line = RWND size.

thousands of test results. As argued in previous sections, multiple parallel TCP connections shouldn't lead to higher throughput than tests with a single connection.

What can be impacted is the initial phase (in our example ≈ 1.5 s) before the connection ramps-up. With higher number of connections we expect faster ramp-up. For certain number of connections which doesn't need to be very high (in our example 5) the ramp-up duration is negligible. We have also seen that more connections fill the buffers along the path faster.

1.7 Comparison of iPerf3, HTTP and FTP Throughput

Since the above cited IETF standards reference iPerf and the 3GPP standard recommends HTTP and FTP for measuring achievable throughput, we set up simple measurement to compare throughput differences between these three applications. At the same time we try out different numbers of TCP connections. The expected output (based on the theoretical analysis in previous sections) is that we should observe no difference in average throughput for different numbers of connections if the measurement is long enough, so that the different ramp up phases have negligible impact on the total average throughput. We also expect that there will be no significant difference between HTTP and FTP, as explained in the 3GPP standard.

1.7.1 Setup

Measurements were static. For every application (HTTP, FTP, iPerf3) the test duration was set to 180 s (three times more than recommended in the 3GPP standard). UE was LG F60. Since the measurements were carried out in live LTE network, we cycle through all three applications regularly to assure that tests of every application are spread regularly over the whole day. (As will become clear in chapter 4, section 4.6, it is not good idea to test, e.g., HTTP on Friday and iPerf3 on Saturday.)

Measurements were performed 10.–12.2.2017, ≈ 70 hours. For every application we tested both directions (UL and DL). We collected ≈ 220 measurements per application per direction (DL/UL). For iPerf3 we tried 1, 5 and 10 TCP connections, there we have therefore ≈ 70 measurements per configuration.

1.7.2 Results

For every test we estimated the average throughput. Results are plotted in fig. 1.12 and 1.13. The first empirical CDF indicates that HTTP and FTP really achieve the same throughput in both directions. In UL also iPerf3 achieves the same performance. In DL iPerf3 reaches $\approx 7\%$ higher throughput than HTTP and FTP.

Fig. 1.13 confirms that in long term the different numbers of connections reach the same throughput in both directions (up to some discrepancies around 5 Mbit/s in UL, which may be caused by the fact that we have just ≈ 70 measurements per iPerf3 configuration per direction).

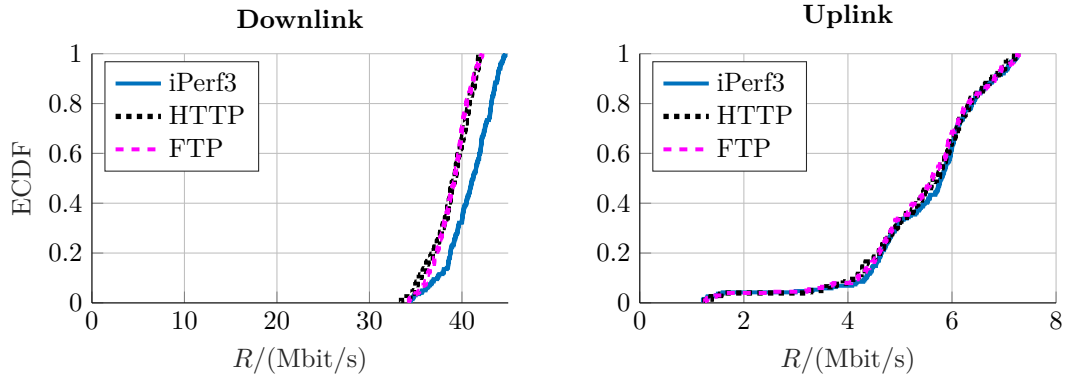


Figure 1.12: Empirical CDF plots of iPerf3, HTTP and FTP throughput in DL and UL. For iPerf3 we took all configurations together here.

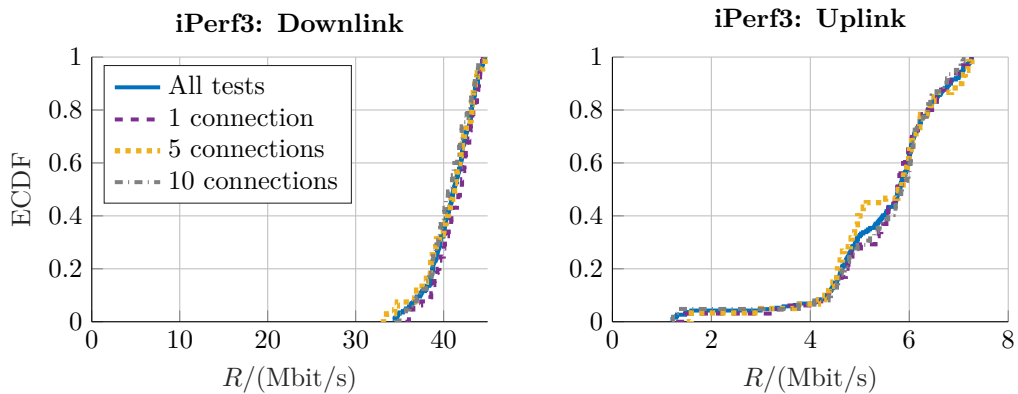


Figure 1.13: ECDF of iPerf3 throughput using different number of TCP connections.

Chapter 2

On the Connection of Data Volumes and Rates in Networks

One of the most important metrics for measuring performance of computer networks is the throughput, in general in bytes over time. We start from scratch defining quantities volume, cumulative volume and rate.

As our focus is on benchmarking in mobile network under test, our goal is to derive a time series of data rate for each user on the same time grid (with highest possible granularity) which will allow us not only to merge TCP connections of single user but also to merge tests of multiple users, e.g. in order to derive current network load at a certain point in time.

Later on we will put these definitions in context of the RTR's open data we are going to use. The concepts we develop are however not limited to throughput only, therefore we formulate the problems and the notation as general as possible.

In the rest of this chapter we discuss specific problems, which occur in RTR's open data: thinning, presence of oscillations and presence of traffic shaping.

2.1 From Volume to Rate

2.1.1 Basic Definitions

Definition 2.1.1 (The true time-continuous rate $r(t)$). Real function $r(t): \mathbb{R} \rightarrow \mathbb{R}_0^+$, which is bounded:

$$\begin{aligned} r_{\min} &\leq r(t) \leq r_{\max} \quad \forall t \in \mathbb{R}, \\ r_{\min} &= \inf_t r(t), \quad r_{\max} = \sup_t r(t). \end{aligned}$$

Note: The most of the theory can be easily extended to $r(t): \mathbb{R} \rightarrow \mathbb{R}$. In the following chapters we are however interested only in the nonnegative rates, therefore we limit the image of the function to the set of nonnegative real numbers.

Definition 2.1.2 (The time-location of the k -th sample: $t[k]$).

$$t[k] \in \mathbb{R} \quad \wedge \quad t[k-1] < t[k] < t[k+1] \quad \forall k \in \mathbb{Z}.$$

Definition 2.1.3 (The k -th time interval $\mathcal{I}[k]$ and its size $|\mathcal{I}[k]|$).

$$\mathcal{I}[k] \triangleq (t[k-1], t[k]], \quad |\mathcal{I}[k]| \triangleq t[k] - t[k-1].$$

Definition 2.1.4 (Binning). Let $r[k]$ denote the k -th bin of size $|\mathcal{I}[k]|$. The bin replaces all values of $r(t)$ within the argument interval $t \in \mathcal{I}[k]$ by their mean:

$$r[k] \triangleq \frac{1}{|\mathcal{I}[k]|} \int_{\mathcal{I}[k]} r(t) dt = \frac{1}{t[k] - t[k-1]} \int_{t[k-1]}^{t[k]} r(t) dt.$$

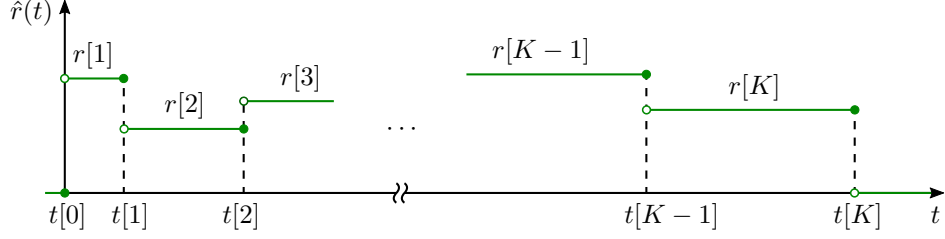


Figure 2.1: $\hat{r}(t)$ is the time-continuous estimate of the true rate $r(t)$. The k -th time discrete sample $r[k]$ represents the average of the true rate in time interval $\mathcal{I}[k] = (t[k-1], t[k]]$.

Definition 2.1.5 (Volume). The total volume in the k -th interval $\mathcal{I}[k]$ is defined as:

$$v[k] \triangleq \int_{\mathcal{I}[k]} r(t) dt = |\mathcal{I}[k]| \cdot r[k].$$

Definition 2.1.6 (The time-continuous estimate of the true rate.). If we know just samples $r[k]$ and not the true rate $r(t)$, we define the time-continuous estimate $\hat{r}(t)$ of the true rate:

$$\hat{r}(t)|_{t \in \mathcal{I}[k]} \triangleq r[k] \quad \forall k \in \mathbb{Z}. \quad (2.1)$$

I.e. the time-continuous function $\hat{r}(t)$ is stepwise constant, for every $t \in \mathcal{I}[k]$ it is equal to the sample value $r[k]$. This definition emphasizes that the value $r[k]$ does not characterize just a single time point $t[k]$ but the whole interval $\mathcal{I}[k]$.

The reason for such definition is, that we have no prior knowledge about statistics of $r(t)$, we know only $r[k]$ – the mean of $r(t)$ in interval $\mathcal{I}[k]$. It is therefore reasonable to pick the least informative variant and distribute the volume $v[k]$ uniformly over the whole interval $\mathcal{I}[k]$. The relationship between $\hat{r}(t)$ and $r[k]$ is illustrated in fig. 2.1 for a special case with K samples, i.e. $k \in \{1, \dots, K\}$.

Definition 2.1.7 (Cumulative volume). In special case where $r(t) = 0 \quad \forall t < 0$, we fix $t[0] \triangleq 0$ (from previous definitions then follows $t[k] = 0$ and $v[k] = 0 \quad \forall k \leq 0$) and define cumulative volume samples $w[k]$:

$$w[k] \triangleq \int_0^{t[k]} r(t) dt = \sum_{i=0}^k v[i] = \sum_{i=0}^k |\mathcal{I}[i]| \cdot r[i] = \int_0^{t[k]} \hat{r}(t) dt. \quad (2.2)$$

Note: $w[k] = 0 \quad \forall k \leq 0$. The second, third and fourth equality follow directly from the previous definitions.

Remark. Binning in def. 2.1.4 is a specific way of sampling. It is probably not the best way, but in cases where it occurs we often have no other choice, see examples below.

Example 1 (A weather station measuring precipitation rate $r(t)$). A weather station has a rain gauge which gathers precipitation for a certain time period T and then estimates the precipitation rate in the k -th interval by dividing the total precipitation volume by the period: $r[k] = \frac{1}{T} \int_{(k-1)T}^{kT} r(t) dt$.

Example 2 (Data rate estimation at the receiver.). At irregularly spaced time points $t[k]$ the receiver receives blocks of data of size $v[k]$ bits. We need to characterize the throughput of the connection by some time-continuous function (data rate). Since we know only the sizes of the received blocks and the inter arrival times, it is reasonable to distribute the received volume $v[k]$ uniformly over the corresponding interval: $\hat{r}(t)|_{t \in \mathcal{I}[k]} = v[k]/|\mathcal{I}[k]|$.

2.1.2 Resampling

For implementation purposes we prefer time-discrete rate representation $r[k]$. Because it is problematic to compare two time-discrete sequences with different time-locations of the samples and it is much more convenient to work with equidistantly spaced samples (for example when calculating time-discrete convolution), we need to resample our data at some point. We first present the proposed resampling algorithm and then bring few comments.

Definition 2.1.8 (Resampling). Let $T \in \mathbb{R}^+$ denote the resampling period, $\tilde{r}[k]$ the resampled rate and $\tilde{t}[k] = kT \forall k \in \mathbb{Z}$ the corresponding equidistantly spaced time-locations of the samples. Time intervals after resampling are $\tilde{\mathcal{I}}[k] = (\tilde{t}[k-1], \tilde{t}[k]] = ((k-1)T, kT]$ and their size is constant $|\tilde{\mathcal{I}}[k]| = T$. The values $\tilde{r}[k]$ are chosen such that the following equality holds:

$$\tilde{r}[k] \triangleq \frac{1}{|\tilde{\mathcal{I}}[k]|} \int_{\tilde{\mathcal{I}}[k]} \hat{r}(t) dt = \frac{1}{T} \int_{(k-1)T}^{kT} \hat{r}(t) dt.$$

Definition 2.1.9 (Resampled time-continuous rate $\tilde{r}(t)$). Analogous to def. 2.1.6:

$$\tilde{r}(t)|_{t \in \tilde{\mathcal{I}}[k]} \triangleq \tilde{r}[k] \quad \forall k \in \mathbb{Z}.$$

Remark. If the true rate $r(t)$ is known, i.e. $\hat{r}(t) = r(t)$ (we know all, in general infinitely many, samples), then the resampling in def. 2.1.8 is consistent with the sampling in def. 2.1.4. If $r(t)$ is unknown we use at least its time-continuous estimate $\hat{r}(t)$.

Another benefit of such resampling is, that it preserves average rate in all intervals $t \in [aT, bT]$, where $a, b \in \mathbb{Z}$ and $a \leq b$:

$$\frac{1}{(b-a)T} \int_{aT}^{bT} \tilde{r}(t) dt = \frac{1}{b-a} \sum_{k=a+1}^b \tilde{r}[k] = \frac{1}{(b-a)T} \int_{aT}^{bT} \hat{r}(t) dt. \quad (2.3)$$

The first equality follows from def. 2.1.9, the second one from def. 2.1.8. If it furthermore happens that there are some $i, j \in \mathbb{Z}$ such that $aT = t[i]$ and $bT = t[j]$ than also following holds:

$$\frac{1}{(b-a)T} \int_{aT}^{bT} \tilde{r}(t) dt = \frac{1}{(b-a)T} \int_{aT}^{bT} r(t) dt.$$

In fig. 2.2 we illustrate the resampling of $r[k]$ (i.e. also of corresponding $\hat{r}(t)$). Our resampling corresponds to rebinning which assures that the total volume in each interval $\tilde{\mathcal{I}}[k]$ is the same before and after rebinning.

A more naive approach (direct subsampling from $\hat{r}(t)$, in fig. 2.3) is not a good idea – it has the disadvantage that it can change the total volume in every interval significantly. Example: If a user would perform a data rate test and would apply the direct resampling of $\hat{r}(t)$, the mean data rate of the whole test, which is an important indicator for quality of service (QoS), would be changed – in some cases very significantly.

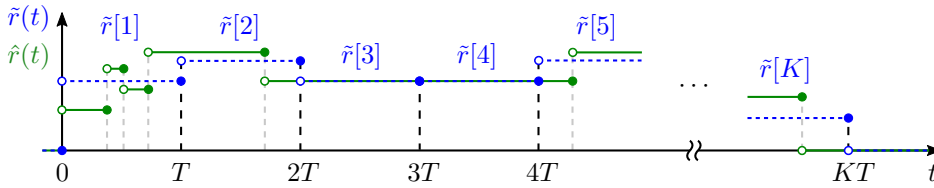


Figure 2.2: “Correct” resampling corresponds to rebinning of the time-continuous estimate $\hat{r}(t)$ in such a way that the mean of resampled rate $\tilde{r}(t)$ is equal to the mean of $\hat{r}(t)$ in every time interval $[(k-1)T, kT]$.

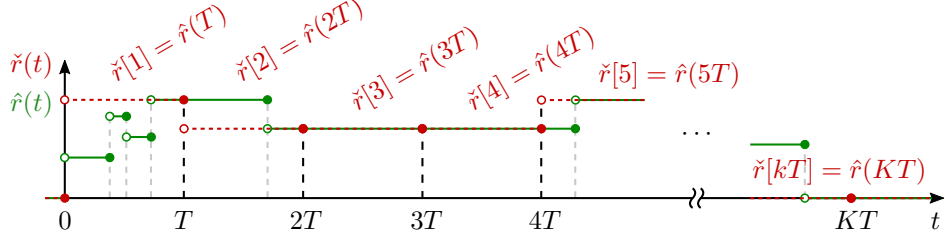


Figure 2.3: An example of “wrong,” direct resampling $\tilde{r}[k] = \hat{r}(kT)$. In this example the mean rate in the interval $[0, T]$ is significantly larger after resampling.

Definition 2.1.10 (Resampled volume).

$$\tilde{v}[k] \triangleq \int_{\tilde{\mathcal{I}}[k]} \tilde{r}(t) dt = T\tilde{r}[k] = \int_{\tilde{\mathcal{I}}[k]} \hat{r}(t) dt.$$

The second equality follows directly from def. 2.1.9, the third equality from def. 2.1.8.

Definition 2.1.11 (Resampled cumulative volume). Posing the same restriction as in def. 2.1.7, i.e. $r(t) = 0 \forall t < 0$ and $t[0] \triangleq 0$ (this leads to $\hat{r}(t) = 0 \forall t < 0$ and thus $\tilde{r}[k] = 0 \forall k \leq 0$), we define resampled cumulative volume:

$$\tilde{w}[k] \triangleq \int_0^{kT} \tilde{r}(t) dt = \sum_{i=0}^k \tilde{v}[i] = T \sum_{i=0}^k \tilde{r}[i].$$

Theorem 1. *Resampling of time-discrete rate $r[k]$ (or correspondingly of $\hat{r}(t)$) is equivalent to linear interpolation of cumulative volume $w[k]$.*

Proof. To be consistent with def. 2.1.7 and def. 2.1.11 we pose the same restrictions: $r(t) = 0 \forall t < 0$ and $t[0] \triangleq 0$. Note that this is needed because when we consider cumulative quantity we have to set some time point from which we start to cumulate. We will therefore consider only $k > 0$ and $t[i] > 0$. Note: Any practical physical process has some starting point (e.g. time where we start to measure).

Case 1: For given $k \exists i \in \mathbb{N}$ such that $t[i] = kT$. In this case the new sample location $\tilde{t}[k] = kT$ coincides with an old sample location $t[i]$:

$$\tilde{w}[k] = \int_0^{kT} \tilde{r}(t) dt = \int_0^{kT} \hat{r}(t) dt = \int_0^{t[i]} \hat{r}(t) dt = w[i].$$

The second equality follows from eq. (2.3), the third from fact that $t[i] = kT$, the fourth and the first follow from definitions of cumulative volume and resampled cumulative volume.

Case 2: For given k there is no $i \in \mathbb{N}$ fulfilling $t[i] = kT$, i.e. $t[i] \neq kT \forall i \in \mathbb{N}$. In this case we can find some $j \in \mathbb{N}$ such that $t[j-1] < kT < t[j]$. I.e. the new sample location lies between two old sample locations. Then it follows:

$$\tilde{w}[k] = \underbrace{\int_0^{t[j-1]} \hat{r}(t) dt}_{w[j-1]} + \underbrace{\int_{t[j-1]}^{kT} \hat{r}(t) dt}_{(kT-t[j-1]) \cdot r[j]} = w[j-1] + \frac{kT - t[j-1]}{t[j] - t[j-1]}(w[j] - w[j-1]). \quad (2.4)$$

In the first equality we just reuse second equality from case 1, split the integral and use $w[j-1] = \int_0^{t[j-1]} \hat{r}(t) dt$ and the fact that $\hat{r}(t) = r[j] \forall t \in (t[j-1], t[j]]$. In the second equality we express $r[j]$ as $\frac{v[j]}{|\mathcal{I}[j]|} = \frac{w[j] - w[j-1]}{t[j] - t[j-1]}$. \square

2.1.3 Multiple Connections: Merging

In this subsection we will consider rate consisting of $C \in \mathbb{Z}^+$ connections (e.g. data rate and multiple parallel TCP connections, data rate and multiple UDP streams, rate of fluid flow and multiple pipes, ...). The index $c \in \{1, \dots, C\}$ shall denote the c -th connection.

We extend the notation established in previous two subsections: $r_c(t)$, $t_c[k]$, $r_c[k]$ and $\hat{r}_c(t)$, $v_c[k]$ and $w_c[k]$ are the true rate, sampling times, sampled rate and corresponding time-continuous estimate, volume and cumulative volume of the c -th connection. The bounds of the true rate may differ for different connections: $r_{\min,c} \leq r_c(t) \leq r_{\max,c} \forall t$.

Similarly for the resampled quantities: $\tilde{t}_c[k]$, $\tilde{r}_c[k]$ and $\tilde{r}_c(t)$, $\tilde{v}_c[k]$ and $\tilde{w}_c[k]$ are the resampling time-location, resampled rate samples and corresponding time-continuous function, resampled volume and cumulative volume of the c -th connection. Note that $\tilde{t}_c[k] = \tilde{t}[k] = kT$, i.e. for all connections we use the same resampling time-locations.

Definition 2.1.12. The total true rate $r(t)$ is given by the sum of true rates of individual connections:

$$r(t) = \sum_{c=1}^C r_c(t).$$

Note: $r(t)$ as well as every $r_c(t)$ separately fulfill all the previous relationships. This definition builds an additional relation between the total true rate $r(t)$ and true rates of individual connections $r_c(t)$.

Merging of Cumulative Volume Sequences

Let's assume that $K_c \in \mathbb{Z}^+$ cumulative volume samples are given for each connection, i.e. $(t_c[k], w_c[k])_{k \in \{1, \dots, K_c\}}$ is known $\forall c \in \{1, \dots, C\}$. We require that rate $r_c(t)$ is zero for $t < 0$ for all connections, thus we can formally set $t_c[0] \triangleq 0$, $w_c[0] \triangleq 0 \forall c$. Since the last known sample is $(t_c[K_c], w_c[K_c])$, we assume $r_c(t) = 0$ for $t > t_c[K_c]$ and formally define additional cumulative volume sample $w_c[K_c + 1] \triangleq w_c[K_c]$ with time location, which is larger than any given time location $t_c[K_c + 1] > \max_{c'} \{t_{c'}[K_{c'}]\}$ but otherwise arbitrary.

After this preparation we can formulate our problem: How to represent the total rate if only samples $(t_c[k], w_c[k])$ are given for every connection? We propose the following merging algorithm.

Definition 2.1.13 (Merging algorithm).

$$\hat{r}(t) = \sum_{c=1}^C \hat{r}_c(t). \quad (2.5)$$

Theorem 2. Cumulative volume samples obtained by the merging algorithm in eq. (2.5) are given by sum of linearly interpolated cumulative volumes of individual connections

$$w[k] = \sum_{c=1}^C \left(w_c[i_c] + \frac{t[k] - t_c[i_c]}{t_c[i_c + 1] - t_c[i_c]} (w_c[i_c + 1] - w_c[i_c]) \right)$$

with time locations

$$\{t[1], \dots, t[K]\} = \{t_1[1], \dots, t_1[K_1]\} \cup \{t_2[1], \dots, t_2[K_2]\} \cup \dots \cup \{t_C[1], \dots, t_C[K_C]\},$$

$$t[1] < t[2] < \dots < t[K],$$

where $k \in \{1, \dots, K\}$ and $i_c \in \{0, \dots, K_c\}$ such that $t_c[i_c] < t[k] \leq t_c[i_c + 1]$.

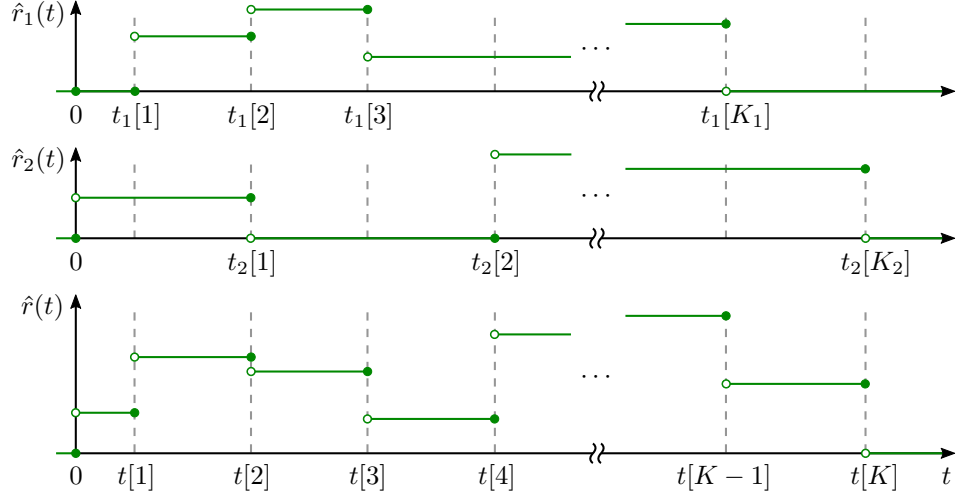


Figure 2.4: Merging rates of two connections: $\hat{r}(t) = \hat{r}_1(t) + \hat{r}_2(t)$. First connection can be represented by K_1 pairs $(t_1[k], r_1[k])$, second connection by K_2 pairs $(t_2[k], r_2[k])$ and the merged sequence by K pairs $(t[k], r[k])$.

Proof. The merging algorithm in def. 2.1.13 is illustrated in fig. 2.4 for the case of two connections. We immediately see, that sample locations of the merged sequence are given by time points where discontinuities of $\hat{r}(t)$ occur, i.e. at time points

$$\{t[1], \dots, t[K]\} = \{t_1[1], \dots, t_1[K_1]\} \cup \{t_2[1], \dots, t_2[K_2]\} \cup \dots \cup \{t_C[1], \dots, t_C[K_C]\} \quad (2.6)$$

which are sorted as follows:

$$t[1] < t[2] < \dots < t[K].$$

Because $t_c[0] = 0$, $w_c[0] = 0 \forall c$, we can formally set also $t[0] = 0$, $w[0] = 0$.

The cumulative volume values $w[k]$ can be calculated according to the last expression of eq. (2.2), then we apply eq. (2.5) and exchange the summation and integration. For a given $t[k]$, $k \in \{1, \dots, K\}$, we can find for every connection an index $i_c \geq 0$ such that $t_c[i_c] < t[k] \leq t_c[i_c + 1]$ and split the integral into two parts. Then we apply eq. (2.2) to the first integral. For the second integral we use eq. (2.1) which tells us that $\hat{r}_c(t)$ is constant and equal to $r_c[i_c + 1]$ in the whole interval $\mathcal{I}_c[i_c + 1] = (t_c[i_c], t_c[i_c + 1]]$. Finally we rewrite the rate sample $r_c[i_c + 1]$ as $\frac{w_c[i_c + 1] - w_c[i_c]}{t_c[i_c + 1] - t_c[i_c]}$:

$$\begin{aligned} w[k] &= \int_0^{t[k]} \hat{r}(t) dt = \sum_{c=1}^C \int_0^{t[k]} \hat{r}_c(t) dt = \\ &= \sum_{c=1}^C \left(\underbrace{\int_0^{t_c[i_c]} \hat{r}_c(t) dt}_{w_c[i_c]} + \underbrace{\int_{t_c[i_c]}^{t[k]} \hat{r}_c(t) dt}_{(t[k] - t_c[i_c])r_c[i_c + 1]} \right) = \\ &= \sum_{c=1}^C \left(w_c[i_c] + \frac{t[k] - t_c[i_c]}{t_c[i_c + 1] - t_c[i_c]} (w_c[i_c + 1] - w_c[i_c]) \right) \end{aligned} \quad (2.7)$$

The final expression is a sum of linearly interpolated cumulative volume sequences of individual connections.

Note: From eq. (2.6) follows $t[K] = \max_c \{t_c[K_c]\}$. The index i_c which fulfills $t_c[i_c] < t[k] \leq t_c[i_c + 1]$ always exists for $k \in \{1, \dots, K\} \forall c$ because we set $t_c[0] = 0$ and $t_c[K_c + 1] > \max_{c'} \{t_{c'}[K_{c'}]\}$ for every connection c . In cases where $t[k] = t_c[i_c + 1]$, the second summand in eq. (2.7) disappears. \square

2.1.4 Resampling and Merging Combined

Theorem 3. *Merging and resampling commute.*

Proof. Recall that $\tilde{w}[k] = \int_0^{kT} \tilde{r}(t) dt$ and that resampling is characterized by following equation (follows from eq. (2.3)):

$$\int_0^{kT} \tilde{r}(t) dt = \int_0^{kT} \hat{r}(t) dt.$$

Now we can write:

$$\begin{aligned} \tilde{w}[k] &= \int_0^{kT} \tilde{r}(t) dt = \int_0^{kT} \hat{r}(t) dt = \int_0^{kT} \left(\sum_{c=1}^C \hat{r}_c(t) \right) dt = \\ &= \sum_{c=1}^C \int_0^{kT} \hat{r}_c(t) dt = \sum_{c=1}^C \int_0^{kT} \tilde{r}_c(t) dt = \sum_{c=1}^C \tilde{w}_c[k]. \end{aligned} \quad (2.8)$$

Since $\tilde{r}[k] = (\tilde{w}[k] - \tilde{w}[k-1])/T$ we immediately obtain from eq. (2.8) also

$$\tilde{r}[k] = \sum_{c=1}^C \tilde{r}_c[k]. \quad (2.9)$$

On the left hand side we have resampled version of merged rate, on the right hand side merged version of resampled rates. \square

We've just proved that the two block diagrams in fig. 2.5 are equivalent. The merging operation in the diagram b) is defined by eq. (2.5) and can be expressed in terms of cumulative volume in closed form as in eq. (2.7). At the input of both block diagrams there are sequences of two-tuples because we have to specify also time locations which are in general different for every connection and not equidistantly spaced. At the output there is a sequence of just "one-tuples," because the index k is sufficient in order to determine the corresponding time location $t = kT$.

Because in this subsection we consider merging and resampling together for the first time, we have to distinguish between K , the number of samples after merging without resampling, and \tilde{K} , the number of samples after resampling. We don't have to "store" index $k = 0$ because in our assumptions all samples are zero at time $t = 0$. Also after the last sample the rate is assumed to be zero (or unknown).

Different representation of algorithms a) and b) is possible in terms of stepwise continuous rates $\hat{r}_c(t)$, $\tilde{r}_c(t)$, $\hat{r}(t)$ and $\tilde{r}(t)$ – fig. 2.6. Every time-continuous rate is equivalent to corresponding discrete rate $r_c[k]$, $\tilde{r}_c[k]$, $r[k]$ and $\tilde{r}[k]$ and corresponding cumulative volume sequence. The resampling of rate is performed in such a way that mean rate in interval $[aT, bT]$ is same before and after resampling for every $a, b \in \mathbb{Z}, a < b$, as follows from eq. (2.3) (which follows from def. 2.1.8 and 2.1.9).

2.1.5 Summary

In this section we established a time-continuous representation of rate based on limited number of volume or cumulative volume samples (measurements). We proposed an algorithm for resampling the rate and for merging rates of multiple connections. We have shown that both algorithms commute.

We derived closed form expressions for cumulative volume after resampling in eq. (2.4) and after merging in eq. (2.7). Thank to these expressions we need no time-continuous rate – for implementation purposes we prefer resampled discrete cumulative volume sequences.

The time-continuous rates however helped us to build an intuition behind the merging and resampling and in some cases they are mathematically more convenient – e.g. in eq. (2.5).

Note: There are other possibilities how to merge multiple connections, e.g. [38]. In appendix A.1 we introduce this alternative merging algorithm. In appendix A.2 we compare both merging algorithms and argue that the one we proposed is based on more reasonable assumptions.

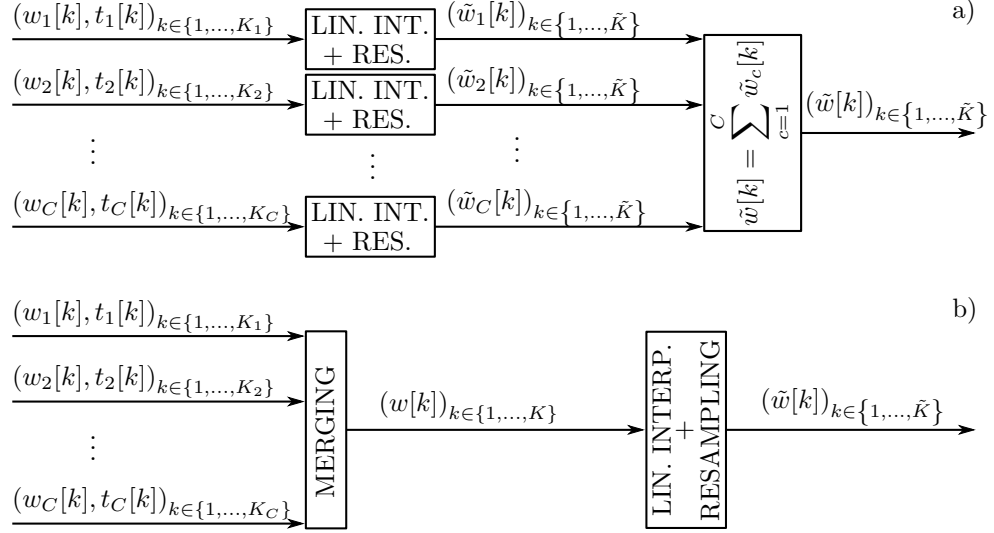


Figure 2.5: Two equivalent algorithms for calculating the total resampled cumulative volume sequence. The merging is defined by eq. (2.5) and can be expressed as in eq. (2.7).

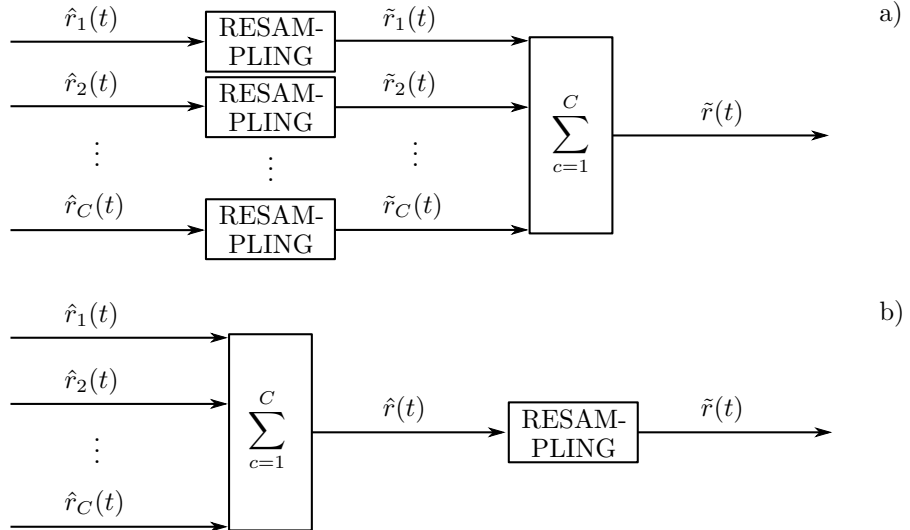


Figure 2.6: Representation of algorithms from fig. 2.5 in terms of stepwise continuous rates. Resampling of rate $\hat{r}(t) \rightarrow \tilde{r}(t)$ is given by def. 2.1.8 and 2.1.9. For time-continuous rates the merging operation is replaced by summation, which is convenient for mathematical description but not suitable for discrete implementation.

2.2 Smoothing: Reducing Uncertainty of Binned Data

The true rate $r(t)$ is unknown. In this section we assume only equidistant sampling at times $\tau + kT$ where τ is a constant offset. We also assume that we can obtain only one bin sequence with one given offset τ .

Example 3 (Weather station ctd.). Let us recall the example 1 with the weather station measuring precipitation rate. The problem is that we obtain different sequences ${}^\tau r[k]$ for different offsets τ , i.e., it does matter whether we start to measure at 8:01 or at 8:03 if the sampling period is, e.g., $T = 5$ min.

As we will see later, the worst case per-bin-average-error (or uncertainty) compared to all other possible shifts $\tau \in \left(-\frac{T}{2}, \frac{T}{2}\right]$ can be expressed as $\varepsilon_{wc} = \frac{r_{\max} - r_{\min}}{2}$. In case where $r_{\min} = 0$ this means that if we observe a bin with maximum precipitation rate of ${}^\tau \hat{r} = \max_k {}^\tau r[k]$, we understand that maximum 5-minute-average of $r(t)$ was between ${}^\tau \hat{r}$ and $2 \cdot {}^\tau \hat{r}$.

2.2.1 Definitions

Let us remind that according to def. 2.1.1 the true rate $t(t)$ is bounded:

$$r_{\min} \leq r(t) \leq r_{\max} \quad \forall t \in \mathbb{R}.$$

In what follows, for sake of simplicity, we specialize the definition 2.1.4 to an equidistant binning – i.e. $t[k-1] = \tau + (k-1)T$ and $t[k] = \tau + kT$:

Definition 2.2.1 (Equidistant Binning). Let ${}^\tau r[k]$ denote the k -th bin of size T with an offset τ . The bin replaces all values of $r(t)$ in the time-interval $t \in (\tau + (k-1)T, \tau + kT]$ by their mean:

$${}^\tau r[k] \triangleq \frac{1}{T} \int_{\tau+(k-1)T}^{\tau+kT} r(t) dt,$$

$$T \in \mathbb{R}^+, \quad \tau \in \left(-\frac{T}{2}, \frac{T}{2}\right], \quad k \in \mathbb{Z}.$$

Definition 2.2.2 (Time-continuous representation of bin sequence).

$${}^\tau r(t)|_{t \in (\tau+(k-1)T, \tau+kT]} \triangleq {}^\tau r[k] \quad \forall k \in \mathbb{Z}.$$

Remark. The reason why we limit the shift to $\tau \in \left(-\frac{T}{2}, \frac{T}{2}\right]$ is, that any shift $\tau \in \mathbb{R}$ can be written as $\tau + lT$, $l \in \mathbb{Z}$, which leads to: ${}^{\tau+lT} r[k] = {}^\tau r[k+l]$ (follows directly from def. 2.2.1). Only indexing is different, the shape of bins is the same, both sequences lead to identical time-continuous representation ${}^\tau r(t)$.

Definition 2.2.3 (Per-bin average uncertainty).

$${}^{\tau_1, \tau_2} \varepsilon[k] \triangleq \frac{1}{T} \int_{\tau_1+(k-1)T}^{\tau_1+kT} |{}^{\tau_1} r(t) - {}^{\tau_2} r(t)| dt = \frac{1}{T} \int_{\tau_1+(k-1)T}^{\tau_1+kT} |{}^{\tau_1} r[k] - {}^{\tau_2} r(t)| dt.$$

The second equality follows from the fact that ${}^{\tau_1} r(t)$ is constant and equal to ${}^{\tau_1} r[k]$ in the whole integration interval according to def. 2.2.2.

Remark. The value ${}^{\tau_1, \tau_2} \varepsilon[k]$ characterizes average difference between ${}^{\tau_1} r(t)$ and ${}^{\tau_2} r(t)$ at the time interval corresponding to bin ${}^{\tau_1} r[k]$. I.e. it tells us what is the average uncertainty (or error) of the k -th bin obtained with offset τ_1 compared to binning with a different offset τ_2 . Note that ${}^{\tau_1, \tau_2} \varepsilon[k] \neq {}^{\tau_2, \tau_1} \varepsilon[k]$.

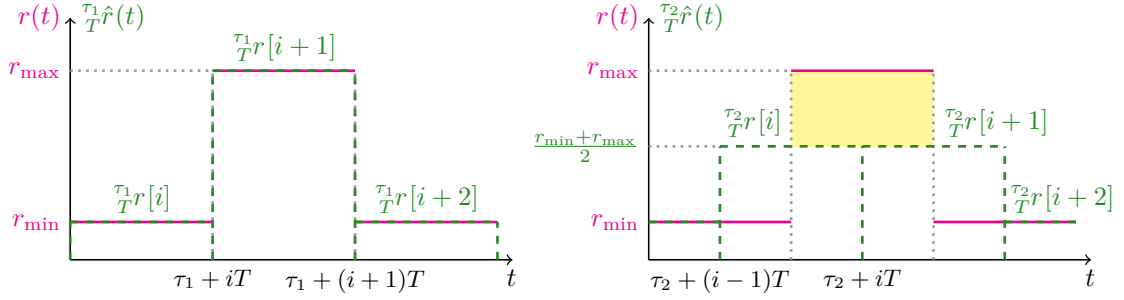


Figure 2.7: Illustration showing the worst case error. Left: binning of the true rate $r(t)$ with offset τ_1 . Right: binning with offset τ_2 . Bin size T is constant but otherwise arbitrary. We obtain the worst case error for $|\tau_1 - \tau_2| = T/2$. The yellow area corresponds to $T \cdot \tau_1, \tau_2 \varepsilon[i+1]$. Thus $\tau_1, \tau_2 \varepsilon[i+1] = \frac{r_{\max} - r_{\min}}{2}$.

2.2.2 Worst Case Uncertainty

Theorem 4 (Worst case uncertainty). *The worst case uncertainty (where uncertainty is defined by the def. 2.2.3) is $\varepsilon_{\text{wc}} = \frac{r_{\max} - r_{\min}}{2}$ and it does not depend on the bin size T .*

Proof. We obtain the largest average difference in a particular bin if we consider situation illustrated in fig. 2.7, i.e. for first offset τ_1 we choose such bin locations that whole i -th bin is filled with the minimum true rate r_{\min} and the whole $(i+1)$ -th bin is filled with the maximum true rate r_{\max} . The largest per-bin average difference is then obtained for τ_2 such that $|\tau_1 - \tau_2| = \frac{T}{2}$, i.e. when the center of a bin corresponding to sampling with offset τ_2 is located exactly in the middle (its height is therefore $\frac{r_{\min} + r_{\max}}{2}$) – at the location where the true rate jumps from r_{\min} to r_{\max} or vice versa. For the bin $\tau_1 r[i+1]$ we then observe constant difference (compared to shift τ_2) $r_{\max} - \frac{r_{\min} + r_{\max}}{2} = \frac{r_{\max} - r_{\min}}{2}$. \square

Remark. This construct is very artificial and it is not very clear if we will ever encounter such specific true rate $r(t)$ (which is moreover unknown and we can observe only binned samples). More useful metric would be an expected per-bin average error. But since we don't have any prior knowledge about the statistics of the true rate (the only requirement is that it is bounded), we can state only the worst case error.

2.2.3 Smoothing

Definition 2.2.4 (Rectangular window of size n). If n is odd, i.e., $n = 2L + 1$, $L \in \mathbb{Z}_0^+$, we define rectangular window $g_n[l]$ as

$$g_n[l] \triangleq \begin{cases} \frac{1}{n}, & -L \leq l \leq L, \\ 0, & \text{otherwise.} \end{cases} \quad (2.10)$$

If n is even, $n = 2L$, $L \in \mathbb{Z}^+$, we define $g_n[l]$ as

$$g_n[l] \triangleq \begin{cases} \frac{1}{n}, & -L \leq l \leq L - 1, \\ 0, & \text{otherwise.} \end{cases} \quad (2.11)$$

Definition 2.2.5 (Smoothed bin sequence).

$$\tau s^{(n)}[k] \triangleq (\tau r * g_n)[k] = \sum_{l=-\infty}^{\infty} \tau r[k-l] g_n[l]. \quad (2.12)$$

Note: We use the right superscript $^{(n)}$ to denote the size of rectangular window which was used for smoothing. The brackets are included in order to not to confuse the window size with exponentiation. The right subscript can still be used for distinguishing different connections, i.e. $\tau s_c^{(n)}[k] \triangleq (\tau r_c * g_n)[k]$.

Theorem 5.

$$\tau s^{(n)}[kn - L + m] = \tau + mT nT r[k], \quad m \in \{0, 1, \dots, n-1\}, \quad (2.13)$$

where $L = \frac{n}{2}$ if n is even and $L = \frac{n-1}{2}$ if n is odd.

Proof. We start with the definition of the smoothed rate in eq. (2.12)

$$\tau s^{(n)}[k] = \sum_{l=-\infty}^{\infty} \tau r[k-l] g_n[l],$$

and plug in the expression for odd rectangular window (eq. (2.10)) and for even rectangular window (eq. (2.11)):

$$\tau s^{(n)}[k] = \begin{cases} \frac{1}{n} \sum_{l=-L}^L \tau r[k-l] = \frac{1}{nT} \int_{\tau+(k-L-1)T}^{\tau+(k+L)T} r(t) dt, & n \text{ odd}; \\ \frac{1}{n} \sum_{l=-L}^{L-1} \tau r[k-l] = \frac{1}{nT} \int_{\tau+(k-L)T}^{\tau+(k+L)T} r(t) dt, & n \text{ even}. \end{cases} \quad (2.14a)$$

$$\tau s^{(n)}[k] = \begin{cases} \frac{1}{n} \sum_{l=-L}^L \tau r[k-l] = \frac{1}{nT} \int_{\tau+(k-L-1)T}^{\tau+(k+L)T} r(t) dt, & n \text{ odd}; \\ \frac{1}{n} \sum_{l=-L}^{L-1} \tau r[k-l] = \frac{1}{nT} \int_{\tau+(k-L)T}^{\tau+(k+L)T} r(t) dt, & n \text{ even}. \end{cases} \quad (2.14b)$$

The second equality in both, eq. (2.14a) and (2.14b), follows from the definition 2.2.1. Now, when we modify index k to $kn - L + m$, we obtain:

$$\tau s^{(n)}[kn - L + m] = \begin{cases} \frac{1}{nT} \int_{\tau+(kn-2L-1+m)T}^{\tau+(kn+m)T} r(t) dt, & n \text{ odd}; \\ \frac{1}{nT} \int_{\tau+(kn-2L+m)T}^{\tau+(kn+m)T} r(t) dt, & n \text{ even}. \end{cases} \quad (2.15a)$$

$$\tau s^{(n)}[kn - L + m] = \begin{cases} \frac{1}{nT} \int_{\tau+(kn-2L-1+m)T}^{\tau+(kn+m)T} r(t) dt, & n \text{ odd}; \\ \frac{1}{nT} \int_{\tau+(kn-2L+m)T}^{\tau+(kn+m)T} r(t) dt, & n \text{ even}. \end{cases} \quad (2.15b)$$

Let's notice that the upper integration bound is the same for both, eq. (2.15a) and (2.15b). In eq. (2.15a) n is odd, thus in the lower integration bound we obtain $-2L - 1 = -n$. In eq. (2.15b) n is even, i.e. $-2L = -n$ in the lower integration bound. With this substitution also the lower bounds coincide and we can write

$$\tau s^{(n)}[kn - L + m] = \frac{1}{nT} \int_{\tau+mT+(k-1)nT}^{\tau+mT+knT} r(t) dt = \tau + mT nT r[k].$$

for both cases (n even or n odd). The second equality is again just application of def. 2.2.1. \square

Remark. The statement of theorem 5 means that the sequence $\tau s^{(n)}[k]$ contains binning $\tau + mT nT r[k]$, with bin size nT for n different shifts $\tau' \in \{\tau, \tau + T, \dots, \tau + (n-1)T\}$:

$$\begin{aligned} \tau s^{(n)}[kn - L] &= \tau nT r[k], \\ \tau s^{(n)}[kn - L + 1] &= \tau + T nT r[k], \\ &\vdots \\ \tau s^{(n)}[kn - L + n - 1] &= \tau + (n-1)T nT r[k]. \end{aligned}$$

In other words: We increased the bin size n times. But compared to direct binning of $r(t)$ we obtain n sequences $\tau'_T r[k]$ corresponding to n different shifts $\tau' = \tau + mT$. This is illustrated in fig. 2.8

Different shifts $\tau \in \left[-\frac{T}{2}, \frac{T}{2}\right)$ still lead to different sequences $\tau'_T s^{(n)}[k]$ but the worst case error is reduced, as we will see.

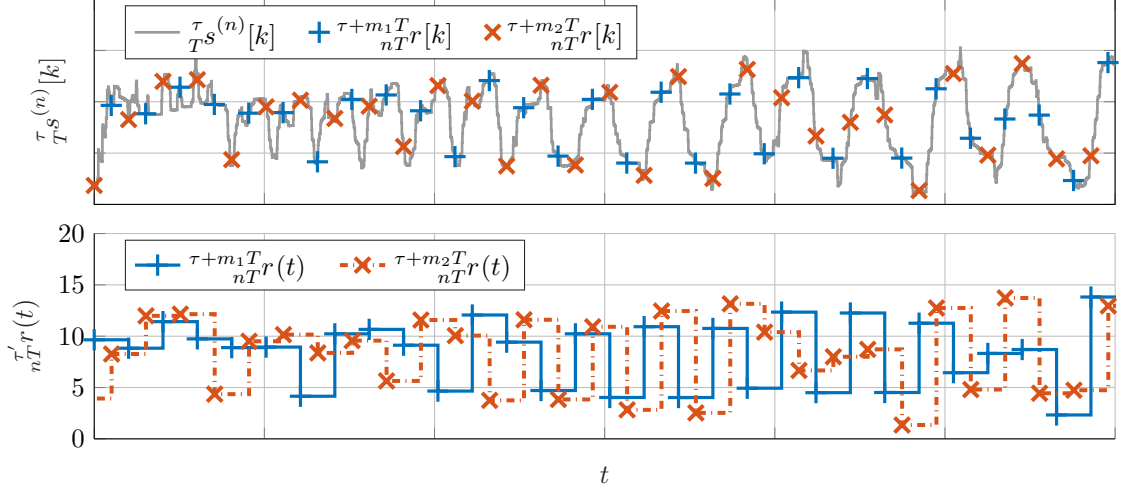


Figure 2.8: The upper plot shows $\tau'_T s^{(n)}[k]$ (gray) – the smoothed version of $\tau'_T r[k]$. By subsampling of $\tau'_T s^{(n)}[k]$ according to eq. (2.13) we obtain n different binnings $\tau'_T r[k]$ with $\tau' \in \{\tau, \tau + T, \dots, \tau + (n-1)T\}$. Here we illustrate two different subsamplings: with $\tau' = \tau + m_1 T$ (blue) and $\tau' = \tau + m_2 T$ (red), where $m_1 = 0$, $m_2 = (n-1)/2$ and $n = 101$. The upper plot shows discrete representations $\tau'_T r[k]$ and the lower plot corresponding time-continuous representations $\tau'_T r(t)$. The samples in the upper plot are aligned with the lower plot such that k -th sample of $\tau'_T s^{(n)}[k]$ corresponds to time $t = kT$ – the time locations of samples in upper plot then correspond to the centers of the time-continuous bins in the lower plot.

Theorem 6 (Reduced worst case uncertainty). *The worst case uncertainty after smoothing with $g_n[l]$ (rectangular window of size n samples) equals $\left(\frac{1}{n} - \frac{1}{2n^2}\right)(r_{\max} - r_{\min})$.*

Proof. We again assume such real rate $r(t)$ and such binning with offset τ_1 and bin size nT , that the bin $\tau_1 r[i+1]$ is whole filled with r_{\max} and bins $\tau_1 r[i]$ and $\tau_1 r[i+2]$ with r_{\min} (fig. 2.9).

Up to this point the approach was the same as the last time. The difference is, that even with larger bin size nT , we get the worst case uncertainty for $|\tau_1 - \tau_2| = T/2$. The reason is, that the smoothed rate $\tau'_T s_n[k]$ contains binnings with all offsets $\in \{\tau, \tau + T, \dots, \tau + (n-1)T\}$. I.e. if τ_2 would equal e.g. $\tau_1 + 3T/4$, the sequence $\tau'_T r[k]$ would be already closer to $\tau_1 + T r[k]$ and would correspond to offset difference of $-T/4$.

The worst case per-bin average uncertainty corresponds to the yellow area in fig. 2.9 divided by the bin duration nT . First we calculate the height of bin $\tau'_T r[i+1]$:

$$\tau'_T r[i+1] = \frac{(n-1/2)T \cdot r_{\max} + T/2 \cdot r_{\min}}{nT} = \frac{(2n-1)r_{\max} + r_{\min}}{2n},$$

and similarly for the bin $\tau'_T r[i]$:

$$\tau'_T r[i] = \frac{(2n-1)r_{\min} + r_{\max}}{2n}.$$

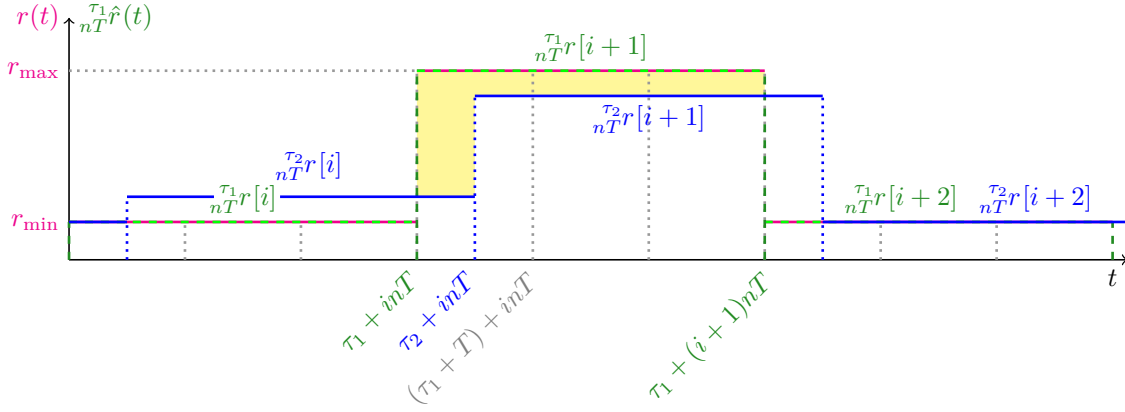


Figure 2.9: Deriving the worst case uncertainty after convolution with rectangular window of size n . Even though the bin duration is now nT , we still observe the worst case uncertainty for the offset difference $|\tau_1 - \tau_2| = T/2$.

The yellow area is then equal to

$$\begin{aligned}
 & (r_{\max} - \frac{\tau_2}{nT}r[i])\frac{T}{2} + (r_{\max} - \frac{\tau_2}{nT}r[i+1])\left(n - \frac{1}{2}\right)T = \\
 & = \frac{(2n-1)r_{\max} - (2n-1)r_{\min}}{2n} \frac{T}{2} + \frac{r_{\max} - r_{\min}}{2n} \left(n - \frac{1}{2}\right)T = \\
 & = (r_{\max} - r_{\min}) \frac{2n-1}{2n} T = (r_{\max} - r_{\min}) \left(1 - \frac{1}{2n}\right)T.
 \end{aligned}$$

Now we divide the yellow area by the bin size nT and obtain the worst case uncertainty after smoothing:

$$\left(\frac{1}{n} - \frac{1}{2n^2}\right)(r_{\max} - r_{\min}).$$

□

Example 4 (Weather station ctd.). In the example 3 we assumed weather station measuring precipitation rate with period $T = 5$ min and some offset τ , $|\tau| \leq 2.5$. With $\hat{r} = \max_k \frac{\tau}{T}r[k]$, we could only say that the maximum 5-minute-average of $r(t)$ was anything between \hat{r} and $2 \cdot \hat{r}$.

If we now convolve the measured sequence $\frac{\tau}{T}r[k]$ with rectangular window $g_6[k]$, we get $\frac{\tau}{T}s^{(6)}[k] = (\frac{\tau}{T}r * g_6)[k]$, which characterizes thirty-minute binnings ($nT = 6 \cdot 5$) with six different offsets $\{\tau, \tau + 5, \dots, \tau + 25\}$. Assuming $r_{\min} = 0$, we obtain the reduced worst case uncertainty $\left(\frac{1}{n} - \frac{1}{2n^2}\right)r_{\max} = \frac{11}{72}r_{\max}$.

I.e. if we observe $\hat{s} = \max_k \frac{\tau}{T}s^{(6)}[k]$, we can say that the maximum 30-minute average of the true rate $r(t)$ was anything between \hat{s} and $\frac{72}{61}\hat{s}$. (In the worst case $\hat{s} = r_{\max} - \frac{11}{72}r_{\max}$.)

2.3 Thinning

In this section we introduce a compression algorithm which we call “thinning.” We don’t claim that it is a good compression algorithm, but we need to analyze it because in later chapters we will have to deal with results delivered by this algorithm.

2.3.1 Thinning Algorithm

Expressed in words: For given cumulative volume sequence and corresponding time locations (in general not equidistant) the thinning algorithm drops minimum necessary number of cumulative volume samples in order to achieve spacing of non-dropped samples $> \Delta_{\min}$, where Δ_{\min} is a constant time interval.

The thinning algorithm is applied for every connection c independently, therefore we omit the lower index c in this section.

Definition 2.3.1 (Thinning algorithm). For a given sequence of pairs $(w[k], t[k])_{k \in \{1, \dots, K\}}$ the thinning algorithm with spacing $\Delta_{\min} \in \mathbb{R}^+$ is a mapping

$$\mathcal{T}: (w[k], t[k])_{k \in \{1, \dots, K\}} \rightarrow (w'[k], t'[k])_{k \in \{1, \dots, K'\}}$$

which fulfills these three conditions:

1. $\forall k \in \{1, \dots, K'\} \exists l \in \{1, \dots, K\}$ such that $(w'[k], t'[k]) = (w[l], t[l])$.
2. $\forall k \in \{2, \dots, K'\}: t'[k] - t'[k-1] > \Delta_{\min}$.
3. K' is maximum number $\in \mathbb{Z}_0^+$ such that 1. and 2. is fulfilled.

Remark. The first condition tells us that the thinned sequence contains only samples which were contained in the original sequence. The second one specifies the spacing. The third one tells us that we shall not drop more sample than necessary, i.e. we drop the minimum of samples such that fist two conditions are fulfilled.

2.3.2 Implementation of Thinning Algorithm

The first sample is $(w[1], t[1])$ with $w[1] > 0, t[1] > 0$. The sample $(w[0], t[0])$ isn't stored anywhere, but it is assumed to be zero at zero time, we make the same assumption for the thinned sequence:

$$t'[0] \triangleq t[0] \triangleq 0, \quad w'[0] \triangleq w[0] \triangleq 0.$$

The first nonzero sample is initialized as follows:

$$t'[1] \triangleq t[1], \quad w'[1] \triangleq w[1].$$

We also initialize two counters i, j and variable t^* holding the last non-dropped sample:

$$i := 2, \quad j := 2, \quad t^* := t[1].$$

Then we repeat following two steps, while $i \leq K$:

- If $(t[i] - t^*) > \Delta_{\min}$:

$$\begin{aligned} t'[j] &\triangleq t[i], \\ w'[j] &\triangleq w[i], \\ t^* &:= t[i], \\ j &+= 1. \end{aligned}$$

Else: do nothing.

- $i += 1$.

Described in words: Given $t'[j-1] = t[i]$, we drop so many samples (choose $t'[j] = t[l]$ with $l > i$ such) that $t'[j] - t'[j-1] > \Delta_{\min}$ for all $j \in \{2, \dots, K'\}$.

2.3.3 Remarks

Given two consecutive samples of thinned sequence $(w'[k], t'[k]) = (w[l+m], t[l+m])$ and $(w'[k-1], t'[k-1]) = (w[l], t[l])$ with some $k, l, m \in \mathbb{Z}^+$, $m > 1$ (i.e. $m-1$ samples were dropped because $t[l+m] - t[l] > \Delta_{\min}$ and $t[l+m-1] - t[l] \leq \Delta_{\min}$) we can express the k -th volume sample of thinned sequence as follows:

$$v'[k] = w'[k] - w'[k-1] = w[l+m] - w[l] = \sum_{i=0}^{l+m} v[i] - \sum_{i=0}^l v[i] = \sum_{i=l+1}^{l+m} v[i]. \quad (2.16)$$

The thinning thus means that we take the total volume measured in the time interval $(t[l], t[l+m])$ and move it to a single sample located at $t[l+m]$. This again shows that it is reasonable to define the rate as a constant during whole interval $(t[l], t[l+m])$ – if we know only total volume in this interval we distribute it uniformly over whole interval.

Comparing Thinned and Non-Thinned Rate

The thinned rate $r'[k] = (w'[k] - w'[k-1]) / (t'[k] - t'[k-1])$ corresponds to resampled version of $r[k]$, because it fulfills our resampling condition (mean rate before and after resampling shall be equal within resampling intervals), which can be shown using eq. (2.16). We again use the time-continuous representation from eq. (2.1):

$$\int_{t'[k-1]}^{t'[k]} \hat{r}'(t) dt = v'[k] = \sum_{i=l+1}^{l+m} v[i] = \int_{t[l]}^{t[l+m]} \hat{r}(t) dt = \int_{t'[k-1]}^{t'[k]} \hat{r}(t) dt.$$

The only problem is that “resampling period” equals

$$t'[k] - t'[k-1] = \Delta_{\min} + e[k-1], \quad \text{with some } e[k-1] \in \mathbb{R}^+,$$

which is in general different for every resampling interval.

Now we choose T such that $\Delta_{\min} = nT$ with some $n \in \mathbb{Z}^+$. If the error is small enough, we obtain

$$t'[k] - t'[k-1] \approx \Delta_{\min} = nT. \quad (2.17)$$

The first idea could be to compare $r'[k]$ with ${}_{nT}\tilde{r}[k]$ (or equivalently $\hat{r}'(t)$ with ${}_{nT}\tilde{r}(t)$), where the left subscript ${}_{nT}$ shall denote the resampled version of $r[k]$ (or $\hat{r}(t)$) with resampling period nT . The problem is that the error e is cumulative:

$$t'[k] = t'[1] + (k-1)\Delta_{\min} + \sum_{i=1}^{k-1} e[i].$$

I.e. even though the first time location would be $t'[1] = qnT_s$ with some $q \in \mathbb{N}$ and all resampling intervals $t'[k] - t'[k-1]$ would be close to nT , the later samples would slowly drift out of the resampling grid $t = knT$.

Noisy Subsampling from the Smoothed Rate

The way out of this is the smoothed rate ${}_Ts^{(n)}[k] = ({}_T\tilde{r} * g_n)[k]$.¹ As explained in subsec. 2.2.3, this smoothed rate represents n rate sequences obtained by resampling of ${}_T\tilde{r}[k]$ by using larger bin size nT with different offsets: ${}_{nT}^0\tilde{r}[k], {}_{nT}^T\tilde{r}[k], \dots, {}_{nT}^{(n-1)T}\tilde{r}[k]$.²

¹Note that we take the resampled version of $r[k]$ because the definition 2.2.5 is valid only for equidistant binning. We also leave out the shift τ for simplicity.

²Please note that here ${}_T\tilde{r}[k]$ is the resampled version of $r[k]$ (resampling period T) and ${}_{nT}^m\tilde{r}[k]$ is the resampled version of ${}_T\tilde{r}[k]$ with offset mT (resampling period nT).

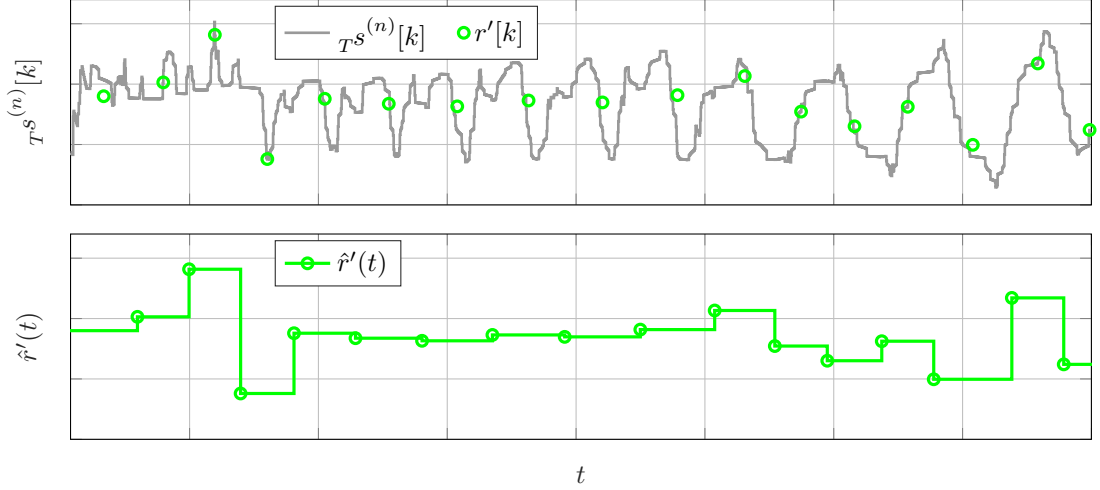


Figure 2.10: Representation of thinned sequence $r'[k]$ as noisy subsampling from $Ts^{(n)}[k]$ (upper plot) and $\hat{r}'(t)$ – time-continuous representation of $r'[k]$ (lower plot).

In ideal case, every sample of $r'[k]$ would be drawn from one of the functions (for every k possibly different function) $\frac{mT}{nT}\tilde{r}[k]$, $m \in \{0, 1, \dots, (n-1)T\}$. The samples $r'[k]$ can of course still be located out of the grid, but now we have n -times finer grid, $t = kT$ instead of $t = knT$.

In other words: Every bin size $t'[k] - t'[k-1]$ is close to nT . With smoothed rate we can compensate for the cumulative time-offset of the thinned bins because we can always pick one of n binnings with bin size nT and offset mT .

We can get rid of the remaining offset $\tau \in \left(-\frac{T}{2}, \frac{T}{2}\right]$ as follows: We introduce

$$t_2 = mT + inT \quad \text{with } m \in \{0, \dots, n-1\}, i \in \mathbb{Z}_0^+ \text{ s.t. } |t_2 - t'[k]| \text{ minimized}, \quad (2.18)$$

then we choose

$$t_1 = mT + (i-1)nT. \quad (2.19)$$

Interval $(t_1, t_2]$ corresponds to bin $\frac{mT}{nT}\hat{r}[i]$ which corresponds to sample $Ts^{(n)}[in - L + m]$ (eq. (2.13)). Thus we replaced the interval $(t'[k-1], t'[k]]$ by the interval $(t_1, t_2]$ which is not the same but hopefully very similar and we write:

$$r'[k] = \frac{mT}{nT}\hat{r}[i] + z[k] = Ts^{(n)}[in - L + m] + z[k], \quad (2.20)$$

where $z[k] \in \mathbb{R}$ is some error.

The idea is that if our assumption in eq. (2.17) is fulfilled, the error shouldn't be too large and we can represent the sequence $r'[k]$ as noisy subsampling from the sequence $Ts^{(n)}[l]$ at locations $l = in - L + m$, as stated in eq. (2.20).

Recall that $L = \frac{n}{2}$ or $L = \frac{n-1}{2}$ depending on the parity of n ; T and n are chosen such that $\Delta_{\min} = nT$ (the larger n and the smaller T the better); and i and m are given by eq. (2.18). This noisy subsampling is illustrated in fig. 2.10.

The goal is actually not to obtain $r'[k]$ from $Ts^{(n)}[l]$ but to reconstruct $Ts^{(n)}[l]$ from $(r'[k], t'[k])_{k \in \{1, \dots, K'\}}$. If we have a model then we can apply for example least squares to obtain $Ts^{(n)}[l]$ from noisy samples.

2.4 Oscillations

2.4.1 Model

In this section we consider a special case of rates of individual connections. Let's assume that the total rate consists of three connections: $\hat{r}(t) = \sum_{c=1}^3 \hat{r}_c(t)$. Further we assume resampling with period T , i.e. $\tilde{r}(t) = \sum_{c=1}^3 \tilde{r}_c(t)$ which is equivalent to $\tilde{r}[k] = \sum_{c=1}^3 \tilde{r}_c[k]$. And finally smoothing with rectangular window of size n . Because convolution is linear operation, we can write

$$_T s^{(n)}[k] = \sum_{c=1}^3 _T s_c^{(n)}[k]. \quad (2.21)$$

Now we model $_T s_c^{(n)}[k]$ as an exponential chirp with frequency function

$$f(t) = \alpha + \beta e^{-\gamma t}, \quad \alpha, \beta, \gamma \in \mathbb{R}_0^+, \quad (2.22)$$

which corresponds to phase function

$$\phi(t) = \phi_0 + 2\pi \int_0^t f(\tau) d\tau = \phi_0 + 2\pi \left(\alpha t - \frac{\beta}{\gamma} (e^{-\gamma t} - 1) \right). \quad (2.23)$$

Finally we require that $_T s_c^{(n)}[k]$ has support $\{0, \dots, K\}$ with K even and we thus obtain

$$_T s_c^{(n)}[k] = (a + b \cdot \sin(\phi(kT))) \cdot \text{rect}\left[k - \frac{K}{2}; \frac{K}{2}\right], \quad \forall k, \quad (2.24)$$

where $a, b \in \mathbb{R}^+$, $a > b$ and

$$\text{rect}[k; L] \triangleq \begin{cases} 1, & |k| \leq L; \\ 0, & \text{otherwise.} \end{cases}$$

We choose arbitrarily $\phi_0 = 0$ and for the frequency function we use $\alpha = 2.254$, $\beta = 9.605$, $\gamma = 0.5331$ (this will become clear in sec. 4.4, chapter 4), further we have $T = 1$ ms and $K = 7000$. Constant a represents DC part of the signal $_T s_c^{(n)}[k]$ and constant b scales the amplitude. For our later problem statement these two constants are not important, in fig. 2.11 we chose $a = 6$, $b = 5$.

2.4.2 Spectrogram

Since the frequency is function of time, we have to use a spectrogram (discrete short time Fourier transform (STFT) [39], [40], [41]) in order to visualize them in the frequency

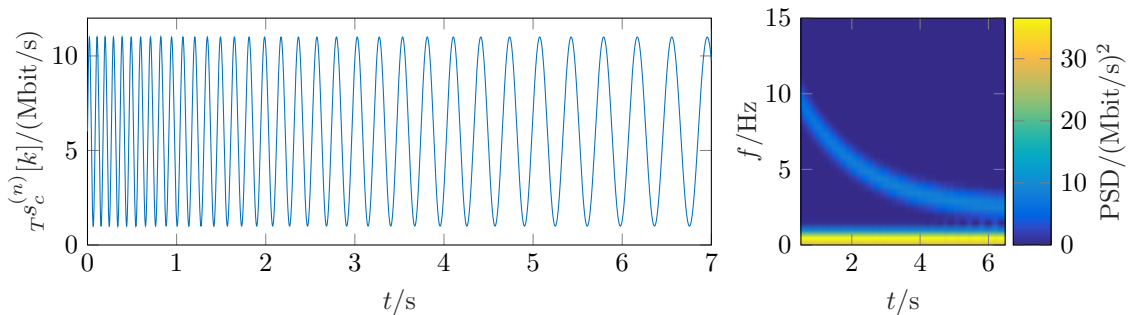


Figure 2.11: The non-zero part of the signal $_T s_c^{(n)}[k] = (a + b \cdot \sin(\phi(kT))) \cdot \text{rect}\left[k - \frac{K}{2}; \frac{K}{2}\right]$ (left) and its spectrogram (right). $T = 1$ ms, $t = kT$.

domain. In figure 2.11 and in what follows in next chapters we used Hamming window of length 1024, overlap of 1023 samples, FFT was calculated always over 2048 samples (in order to get nicer picture with higher resolution) and T was 1 ms.

Since we use DFT, the transformed quantity $\mathcal{F}\{x[i]\}$ has the same unit as $x[i]$, i.e. in case of rate or smoothed rate Mbit/s. Similarly power spectral density (PSD) has unit (Mbit/s)², it is power per sample (observation) not power per Hz. We use following definition of DFT:

$$X[k] = \frac{1}{N} \sum_{n=0}^{N-1} x[n] e^{-2\pi j \frac{nk}{N}}, \quad x[n] = \sum_{k=0}^{N-1} X[k] e^{2\pi j \frac{nk}{N}},$$

which has the advantage that transformed quantity $X[k]$ does not grow with number of samples, for example $A \cos\left(\frac{2\pi m}{N}n\right) \circ \bullet \frac{A}{2} \delta[k-m] + \frac{A}{2} \delta[k+m-N]$, i.e. for one sided representation (real signal, symmetric spectrum) we display just $A \delta[k-m]$, the factor A corresponds to the amplitude of the signal in time domain. For power we have factor $(A/2)^2$ in both terms in case of two-sided representation. In case of one-sided representation we obtain factor $\frac{A^2}{2}$ which corresponds to the power of the sine wave with amplitude A . Note that the definition of DFT above leads to the following form of Parseval's theorem: $\sum_{n=0}^{N-1} |x[n]|^2 = N \sum_{k=0}^{N-1} |X[k]|^2$.

In literature often the following form of the DFT is used $X[k] = \sum_{n=0}^{N-1} x[n] e^{-2\pi j \frac{nk}{N}}$ and $x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{2\pi j \frac{nk}{N}}$.

2.4.3 Reducing Oscillations

First we define following shorthand notation:

Definition 2.4.1.

$$\kappa_{Ts}^{(n)}[k] \triangleq \sum_{c=1}^C Ts_c^{(n)}[k - \kappa_c],$$

with shift-vector $\kappa = (\kappa_1, \dots, \kappa_C)^\top$.

Remark. In whole section we will use only $C = 3$ connections and assume only shifts of the second and the third one, i.e. $\kappa = (0, \kappa_2, \kappa_3)^\top$.

Problem Statement

Let's assume that we want to minimize oscillations of $Ts^{(n)}[k]$ in the interval $t \in [0, KT]$ by shifting $Ts_2^{(n)}[k]$ by κ_2 and $Ts_3^{(n)}[k]$ by κ_3 . Motivation of this step will become clearer in next chapters.

By minimizing oscillations we understand finding shift-vector $\hat{\kappa} = (0, \hat{\kappa}_2, \hat{\kappa}_3)^\top$ such that the variance (power of non-DC components) of the signal in the interval $t \in [0, KT]$, i.e. $k \in \{0, \dots, K\}$ is minimized:

$$\hat{\kappa} = (0, \hat{\kappa}_2, \hat{\kappa}_3)^\top = \arg \min_{(\kappa_2, \kappa_3) \in \mathbb{Z}^2} \frac{1}{K+1} \sum_{k=0}^K \left(\kappa_{Ts}^{(n)}[k] - \left(\frac{1}{K+1} \sum_{k=0}^K \kappa_{Ts}^{(n)}[k] \right) \right)^2. \quad (2.25)$$

Alternatively, if a (minimum-oscillation) reference $s_{\text{ref}}[k]$ is given, we can minimize mean squared difference between $\kappa_{Ts}^{(n)}[k]$ and the reference:

$$\hat{\kappa} = (0, \hat{\kappa}_2, \hat{\kappa}_3)^\top = \arg \min_{(\kappa_2, \kappa_3) \in \mathbb{Z}^2} \frac{1}{K+1} \sum_{k=0}^K \left(\kappa_{Ts}^{(n)}[k] - s_{\text{ref}}[k] \right)^2. \quad (2.26)$$

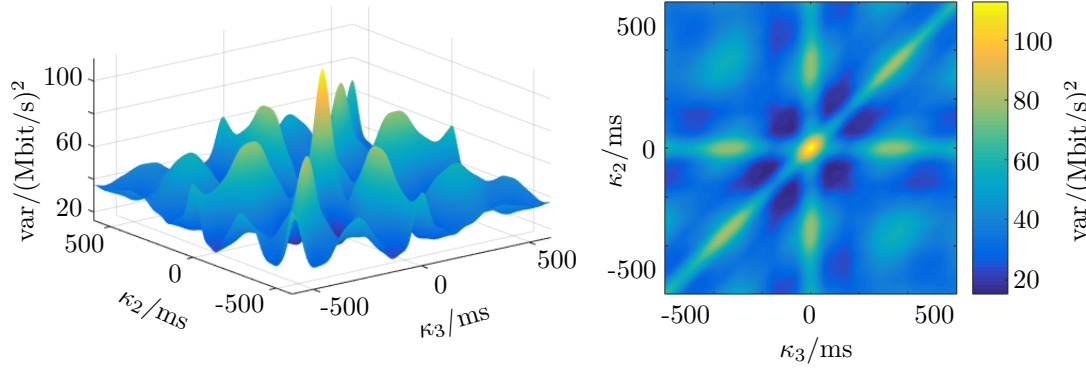


Figure 2.12: Objective function of minimization problem in eq. (2.25) using model from subsec. 2.4.1. Note: var in the axes labels is shorthand for variance of $\hat{\kappa}_T^{(n)}$ calculated for $k \in \{0, \dots, K\}$. With our resampling period $T = 1$ ms, the shift of κ samples corresponds to time shift of κ ms.

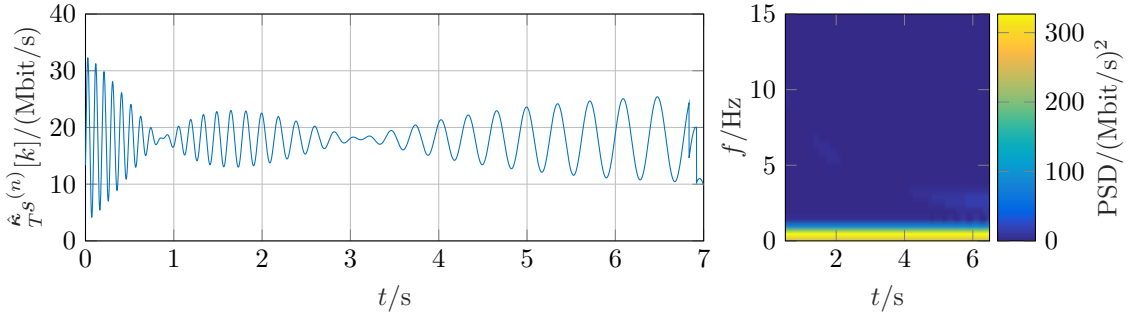


Figure 2.13: Minimum variance solution $\hat{\kappa}_T^{(n)}[k]$ obtained from minimization problem in eq. (2.25) by exhaustive search $(\kappa_2, \kappa_3) \in \{-500, \dots, 500\}^2$.

Brute Force Solution

Since we have no analytic solution for the minimization problem in eq. (2.25), we perform an exhaustive search over all possible shifts $(\kappa_2, \kappa_3) \in \{-500, \dots, 500\}^2$. The objective function is shown in fig. 2.12.

There are six global minima. We pick $\kappa_2 = -81$, $\kappa_3 = -166$ and plot the minimum variance solution $\hat{\kappa}_T^{(n)}[k]$ (fig. 2.13). Oscillations are not removed completely but they are reduced compared to zero shift-vector $\boldsymbol{\kappa} = \mathbf{0}$ which would lead to amplitude $3 \cdot b = 15$. This imperfect solution with remaining oscillations shows certain room for model improvement to better characterize problems in later chapters. However, we will still use the notation and concepts established in this section.

The model is nonlinear, minimization problem is nonconvex and the brute force algorithm has complexity $\mathcal{O}(n^2)$.

Phase Shift

If we consider phase shifts instead of time shifts, oscillations are completely removed for any phase function $\phi(t)$ by choosing shifts $\phi_1 = 0$, $\phi_2 = \frac{2\pi}{3}$, $\phi_3 = 2 \cdot \frac{2\pi}{3}$:

$$e^{j\phi(t)} + e^{j(\phi(t) + \frac{2\pi}{3})} + e^{j(\phi(t) + 2 \cdot \frac{2\pi}{3})} = e^{j\phi(t)} \underbrace{\left(1 + e^{j\frac{2\pi}{3}} + e^{j2\frac{2\pi}{3}}\right)}_{=0} = 0$$

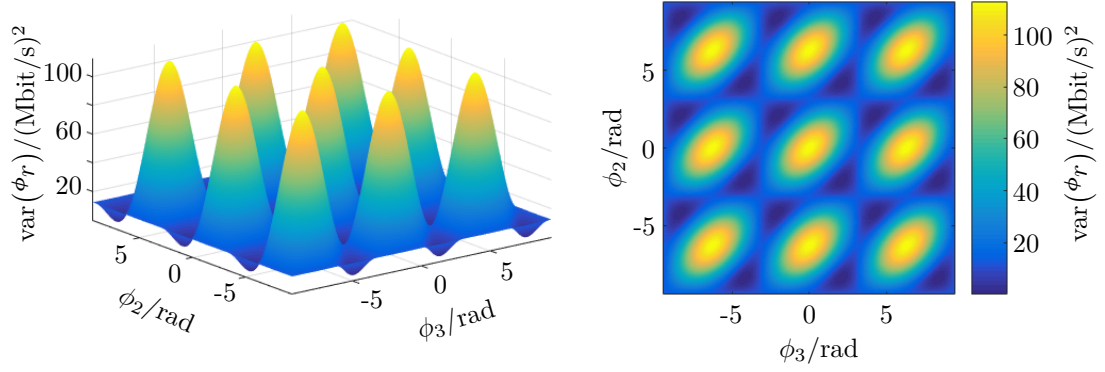


Figure 2.14: Variance of $T^s^{(n)}[k]$ when considering phase shifts ϕ_2, ϕ_3 instead of index shifts κ_2, κ_3 . The phase shift ϕ_1 was fixed to 0. The obtained pattern is 2π -periodic and we would see similar results also for different phase functions $\phi(t)$.

which is well known e.g. from three phase power system, where $\phi(t) = \omega t$. Phase shift solution is not very helpful for us, we just mention it here, because in later chapters we will observe objective function similar to fig. 2.14 (even though we will not consider phase shifts there).

2.5 Extracting Network Performance From User Tests

The limitation of crowdsourced performance measurements as a replacement for drive tests lies in the fact that the UE will show various effects impacting the measurement. One of these effects is the tariff limitation of a standard user. These limitations are typically considering traffic volume per period of time as well as maximum granted data rate. In Austria the latter typically accounts for limits like 10, 20, 40Mbit/s depending on the tariff pricing. Such limits are well below the data rate an LTE link can offer and therefore we might rather measure the user profile than the network performance we are interested in. This would render non-treated results in the RTR's open data useless in such context.

However, a detailed look into the methods of traffic shaping reveals that these methods need a certain time span to evaluate the current data rate of the user, e.g. the first bunch of packets. In the following we used this fundamental property of traffic shaping tools to build and automatized traffic shaping detector. This detector allows us to detect both, the network performance as well as the data rate due to the user profile.

2.5.1 Traffic Shaping Detectors (Existing Method)

A method how to detect traffic shaping is described in [42], it works in cases where token bucket is implemented, e.g. in order to allow a user to exceed the service rate ρ for a limited burst size. The method estimates σ – bucket size (maximum burst size), ρ – token generation rate (shaping rate) and C – link capacity (peak rate).³

The main ideas are illustrated in fig. 2.15: When a user starts to transmit and the token bucket of size σ is full, we first observe the peak rate C , until the bucket is emptied. Then only transmission with token generation rate ρ is possible.

The method relies on detection of beginning (index k_b) and end (index k_e) of level shift in the binned sequence $T^{\hat{r}}[k]$. The peak rate is estimated as the median of all rate bins

³We don't distinguish separate connections in this section, there is therefore no danger of confusing peak rate and total number of connections.

before the level shift

$$\hat{C} = \text{med}_{k \in \{1, \dots, k_b - 1\}} T\hat{r}[k], \quad (2.27)$$

the shaping rate as the median of all rate bins after the level shift

$$\hat{\rho} = \text{med}_{k \in \{k_e + 1, \dots, K\}} T\hat{r}[k], \quad (2.28)$$

and the bucket size as the yellow area in fig. 2.15:

$$\hat{\sigma} = \sum_{k=1}^{k_b} (T\hat{r}[k] - \hat{\rho})T. \quad (2.29)$$

Level Shift Detection

Assuming first n samples of $T\hat{r}[k]$, the index k_b is detected if it fulfills:

1. $k_b \in \{n_L + 1, \dots, n - n_R - 1\}$, $n_L, n_R \in \mathbb{Z}^+$ (enough samples before and after k_b),
2. $T\hat{r}[i] \geq T\hat{r}[j] \quad \forall i \in \{1, \dots, k_b - 1\}, j \in \{k_b + 1, \dots, n\}$,
3. $\text{med}_{i \in \{1, \dots, k_b\}} T\hat{r}[i] > \gamma \text{med}_{j \in \{k_b, \dots, n\}} T\hat{r}[j]$, with suitable threshold γ .

The number n is initialized as $n_L + n_R + 2$, so that in the first iteration we have only one candidate $k_b \in \{n_L + 1\}$. If there is no index fulfilling conditions 1–3, n is increased by one and we proceed with next iteration.

End of level shift, index k_e , is detected in similar way; it is the last index $> k_b$ which fulfills condition 1.

2.5.2 Modified Traffic Shaping Detector

The bin size has to be large enough (in the paper $T = 300$ ms) to suppress noise to such extent that in case of traffic shaping all samples on the left side of k_b are larger than all samples on the right side of k_b (condition 2).

We extend the algorithm [42] by making use of our representation $Ts^{(n)}[k]$ with $T = 1$ ms and different values of n .⁴ In subsec. 2.2.3 we saw that $Ts^{(n)}[k]$ represents n different sequences – binnings $\frac{mT}{nT}r[k]$ with n possible offsets mT , $m \in \{0, \dots, (n - 1)\}$. For every offset we get different sequence (recall fig. 2.8) and therefore different estimates \hat{C} , $\hat{\rho}$, $\hat{\sigma}$. From 301 estimates we should get more information than from single binning with one offset. Measurement results can be found in chapter 4.

Another idea which we haven't tested yet would be to detect level shift jointly, considering all binnings with different offsets together.

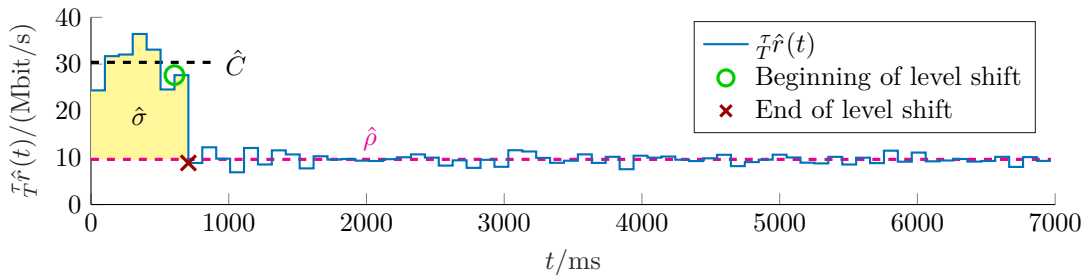


Figure 2.15: An example of traffic shaping. $\hat{C} \approx 30.38$ Mbit/s is the estimate of peak rate, $\hat{\rho} \approx 9.65$ Mbit/s is the estimate of token generation rate and $\hat{\sigma} \approx 14.38$ Mbit ≈ 1.8 MB (the yellow area) is the estimate of the token bucket size.

⁴We prefer odd span of smoothing window because of easier implementation in MATLAB [43].

Chapter 3

Measurement Framework and Tools

The topic of crowdsourced performance measurement for mobile networks can be considered a very young discipline. There exists many different methods and approaches to measure performance in a distributed way. In order to enable big data processing we need to gain a better understanding on how to merge these different data sources into a large set of data, e.g. data rate tests from different countries and regulatory bodies.

In the first part of this chapter we introduce a framework that offers the functionality to compare different tools and configure each of them in a flexible way. Although we already published the concept in [44], there were some changes and improvements implemented. Therefore we present here the whole CMPT framework in its current state.

RTR Mobile Broadband Test (RMBT), the second part of the chapter, plays an essential role because it is the application which produces publicly available crowdsourced RTR's open data we use in the final chapter. It is widely deployed in Austria and its compilations are used also in Slovakia, Slovenia and Serbia (more details in subsection 3.2.4).

3.1 The Measurement Framework (CMPT)

The Crowdsourcing Mobile Performance Tool (CMPT) framework (fig. 3.1) consists of CMPT Android application, user friendly web interface and CMPT server. The CMPT

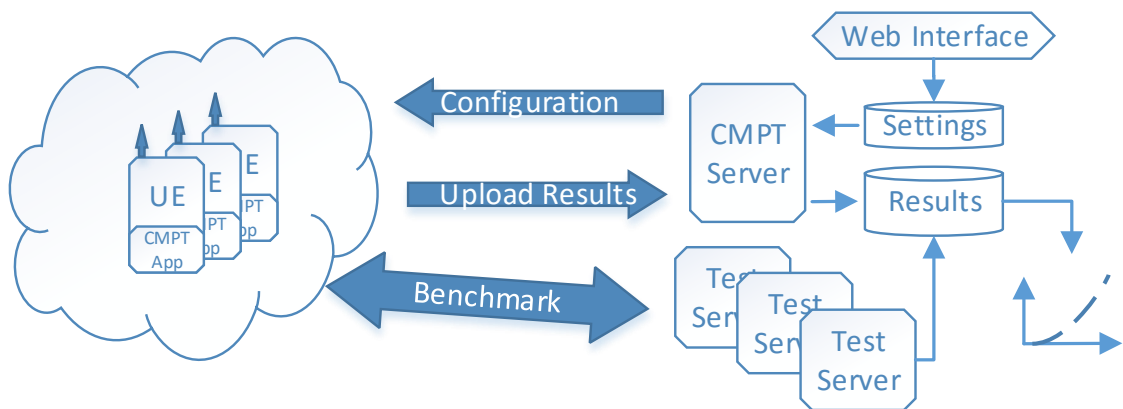


Figure 3.1: The CMPT framework: Configuration is specified via web interface. The UEs running CMPT application obtain configurations from the CMPT server, perform various measurements targeting different servers and upload measurement results back to the CMPT server. Author of this picture: Philipp Svoboda.

server has three components: settings database, results database and script for automatic configuration changes. CMPT further uses applications of third parties (more details in 3.1.4) to perform additional tasks (measurements, data collection).

These applications of third parties may target different test servers which don't have to be under our control (e.g. iPerf3 public test servers [45]). If we target our own test servers, we can merge results from these test servers with CMPT results database.

3.1.1 Android Application

The core of the CMPT Android application is a background service, which keeps running even if no application window is opened. In order to work properly it requires API level 18 or higher, superuser (SU) rights (the device has to be rooted), BusyBox 1.26 or higher [46], and countless permissions (access location, read/write external storage,...). The app was implemented in Java using Android Studio [47] with great assistance of [48] and [49].

Appearance

The CMPT app has four different screens (fig. 3.2 and 3.3): the main screen, list of available WLANs, settings, miscellaneous. Brief overview of each screen follows:

The main screen is divided in three parts. The uppermost part contains a button to start or stop the CMPT background service, list of third parties applications¹ (subsec. 3.1.4) and countdown indicating when which applications is going to be executed (more information in paragraph "Scheduling") if the CMPT background service is running. The middle part contains a statistic showing how many JSON documents with test results were successfully uploaded, how many uploads were unsuccessful, the ID and upload status (ok/failed) of the last JSON document and a button which can reset this statistic. The lower part contains overview of passively collected information (more in paragraph "Passive Monitoring").

The WLAN screen displays a list of available WLANs (if UE's WiFi is turned on) and their RSSI in dBm. This feature is just experimental, in future it can be used to passively

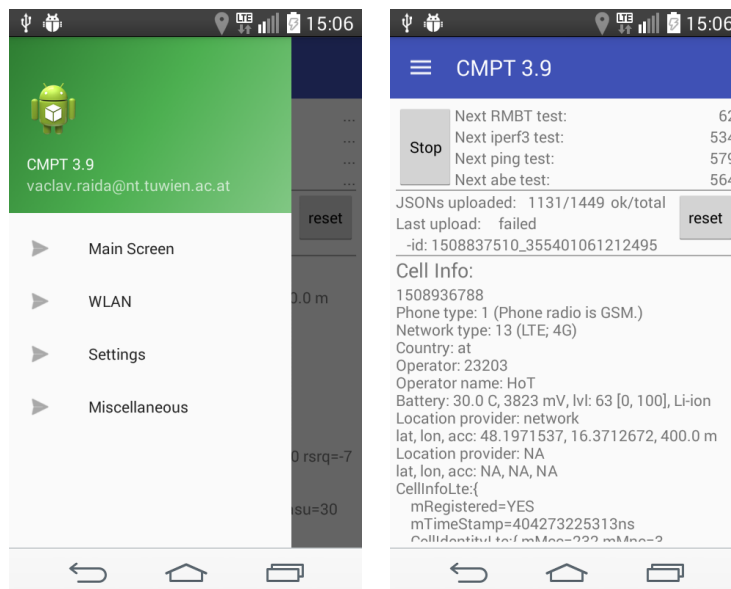


Figure 3.2: CMPT application menu (left) and the main screen (right).

¹The screenshot 3.2 contains an application called ABE (Available Bandwidth Estimation). This is just an old working title for FLARP which is described in subsection 3.1.4.

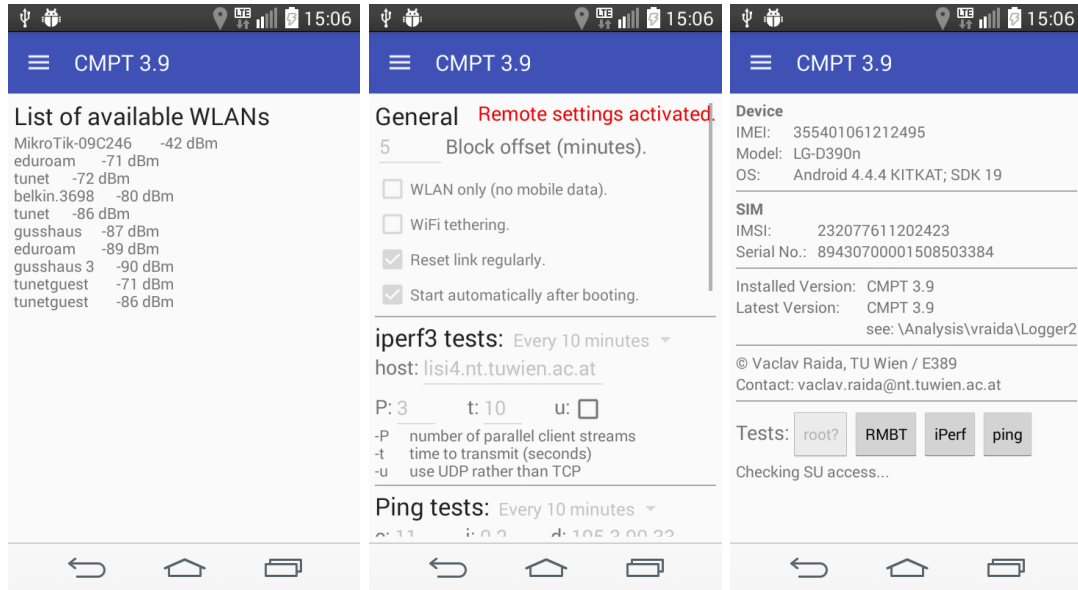


Figure 3.3: CMPT application screens. Left: List of available WLANs (experimental feature). Middle: Overview of test configurations. Right: Miscellaneous.

monitor RSSI levels of available WLANs over time (and upload collected results together with other measurements).

The settings screen shows current configuration of the CMPT and of the third parties applications. In the current version all settings are disabled on the UE side in order to avoid conflicts with remote settings which is described in 3.1.2. In the future version it should be possible to choose on the UE side anytime, whether the local or the remote settings shall be used.

The miscellaneous screen contains basic UE information (IMEI,² device model, Android version and API level), SIM³ information (IMSI,⁴ serial number), information about installed CMPT version and the latest available version, copyright and finally, test buttons. The test buttons are present for debugging, they allow to check SU permissions and to execute third party apps individually without scheduling. The CMPT background service has to be stopped when using test buttons.

Scheduling

In this section we describe how the CMPT application schedules its own tasks together with execution of applications of third parties. CMPT app's tasks are: results upload & configuration download, precharge, final slot and link reset. Third parties' apps we currently use are: ICMP ping, iPerf3, FLARP and Open-RMBT (more details are given in subsection 3.1.4). In the rest of this section we will call the CMPT app's tasks and apps of third parties jointly as "tasks."

The CMPT background service is as an infinite loop which is repeated every second. Scheduling works in 300s blocks, one after another. Beginning of every block is located at time t which is divisible by 300s, i.e. time t such that $t/s \bmod 300 = 0$.

One block is illustrated in fig. 3.4. In every block a fixed-length slot is assigned to every task, such that no slots are overlapping. The length of every slot is hard coded based on requirements of the corresponding task. If the task is going to be executed in current

²International Mobile Equipment Identity

³Subscriber Identity Module

⁴International Mobile Subscriber Identity

Position	Fixed	Task	Slot length
0 s	yes	Precharge	5 s
10 s	no	ICMP ping	15 s
25 s	no	iPerf3	30 s
55 s	no	FLARP	15 s
110 s	no	Results upload, config. download	30 s
140 s	no	Open-RMBT	120 s
275 s	yes	Link reset	—
280 s	yes	Final slot	20 s

Table 3.1: List of CMPT tasks.

block (based on test period, see subsec. 3.1.2, settings), it is started at the beginning of its slot. If any task is not finished at the end of its slot, it is terminated by the CMPT app. The positions of precharge, link reset and final slot are fixed. Positions of all other tasks can be either fixed or randomized, depending on settings.

Table 3.1 summarizes the tasks, their slot lengths and indicates whether every task is fixed or not. The slot of fixed task always starts at time location given by the table. For non-fixed tasks the slot starts either at time given in the table (if randomization is disabled), or at random time (if randomization is enabled).

If randomization is enabled, the scheduling algorithm works like this: In every block there is an available time interval $t \in [5, 275]$ (duration $D = 270$) which is not occupied by any fixed slot. Let's assume there are n non-fixed tasks with the total duration of all their slots d_t seconds. The rest of the available interval will be filled with $n + 1$ random gaps with total duration $d_g = \sum_{i=0}^n d_{g,i} = D - d_t$. We randomly shuffle the order of all n tasks (i.e. decide which one will be the first, second, ..., n -th). We insert a gap between every two neighboring tasks plus one gap before the first task and one gap after the last task. These $n + 1$ gaps are initialized empty. We perform d_g iterations, in every iteration random $i \in \{0, 1, \dots, n\}$ is selected and corresponding $d_{g,i}$ is incremented by one.

- **Results upload, configuration download:** CMPT app connects to the CMPT server, uploads all collected data from the previous 300 s block and downloads configuration for the next 300 s block. HTTPS requests are scheduled using external Volley library [50], which also supports gzip compression to make results uploads and configuration downloads more efficient.
- **Link reset:** If active, the airplane mode is switched on and off in order to disconnect from the mobile network and connect again. The slot doesn't have fixed duration. From our experience it takes not longer than 20 s to connect again. With 5 s reserve, the next 300 s block can start 25 s after the link reset is initiated. The link reset is partially overlapping with the final slot in which no Internet connection is needed.
- **Precharge:** In order to bring connection to an "active" state, e.g. HS-DSCH in UMTS [51] or DL-SCH in LTE [52], a file download is initiated. At the end of the slot the download is interrupted and the downloaded part of the file is deleted.
- **Final slot:** The configuration downloaded in the "UL results, DL conf." slot is checked and if there are any changes, new settings is applied for the next 300 s block. All data and test results collected in the current block are stored to a JSON file and are prepared to be uploaded in the "UL results, DL conf." slot of the next 300 s block. New slot positions, based on new configuration, are generated for the next block.

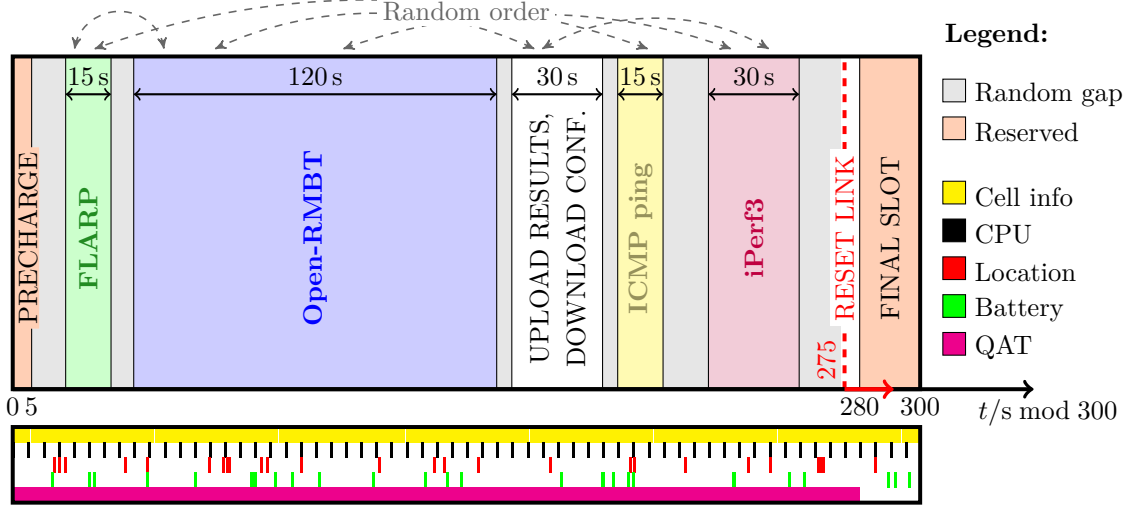


Figure 3.4: CMPT scheduling (upper block) and passive monitoring (lower block).

Passive Monitoring

Together with scheduling of different tasks, CMPT application also persistently collects additional data in passive way, this is illustrated in the lower block of fig. 3.4.

The Android’s cell information is updated every two seconds, however we sample them every seconds in order to don’t miss any sample – even though the infinite loop is supposed to be executed every second, timing is not precise and there is certain drift in order of milliseconds. In case of LTE, cell info includes cell identity (MCC, MNC, CI, PCI, TAC) and signal values (RSRP, RSRQ, RSSNR, CQI, TA) of the cell to which the device is connected as well as of all neighboring cells which are in reach. Analogous values are collected also in case of 2G and 3G.

CPU load and additional information obtained by shell command `busybox uptime` are collected every 5s.

Network location and GPS location as well as battery state (voltage, temperature) are sampled only when the values change – we use Android’s `LocationListener` and `BroadcastReceiver`.

The RSRP values in dBm and RSRQ values in dB are rounded to whole numbers. Since we were interested in signal strength data with finer time and value resolution, Michael Rindler compiled a binary called QAT,⁵ which – once executed – retrieves (in case of LTE) the RSRP and RSRQ values in the cell to which the device is connected. QAT is running for specified time interval with specified sampling period. Results are outputted in JSON format when program finishes. The default sampling period is 300 ms. We kept QAT running in every block from $t = 0$ to $t = 280$ so that the results were ready at the beginning of the final slot.

The QAT feature is experimental and is currently disabled by default, because it is limited to just some of Qualcomm’s chip sets. We used it on LG F60. Comparing values reported by Android and by QAT, we found out that Android values are on average about 0.5 dB higher. Explanation is, that Android drops the decimal part. Since RSRP in dBm and RSRQ in dB are negative, this corresponds to the ceiling operation $\lceil x \rceil$. Indeed, when we took QAT values, performed ceiling operation and than compared two-second averages with the values reported by Android, the 0.5 dB difference disappeared.

⁵Working title derived from qualcommAT, since Michael discovered how to obtain signal strength values from Qualcomm chipset on some devices.

Logged in as **LoggerAdmin**.

Log Out

CMPT – Settings

IMEI: 355401061212461

Device: 355401061212461 – LG F60

General

Offset: Absolute block offset (minutes).

Auto Configuration: ☐ Allow script lisi4.nt.tuwien.ac.at/logger_settings/auto_conf.php

Start Automatically: ☒ Start after boot automatically.

WIFI Tethering: ☐ WIFI hotspot enabled.

Reset Link Regularly: ☒ Airplane mode on and off at the end of every 5-minute block.

WLAN Only: ☐ If check: WIFI on, cellular data off. Vice versa otherwise.

Disable randomization: ☐ Disable random time offsets and random test order.

FLARP

Period: Run test every x minutes (0 = test deactivated). Valid options: 0, 5, 10, 20, 30, 60!

Server:

Port:

Additional: -c: csv output; -j: json output;

iPerf

Period: Run test every x minutes (0 = test deactivated). Valid options: 0, 5, 10, 20, 30, 60!

Duration: seconds

Threads: Number of parallel TCP (default) or UDP flows.

UDP: ☐ Use UDP rather than TCP.

Server:

Port:

OpenRMBT

Period: Run test every x minutes (0 = test deactivated). Valid options: 0, 5, 10, 20, 30, 60!

Duration: seconds

Threads: Number of parallel TCP flows.

Number of Pings:

Server: ID, name or UUID; Currently two: TU-SERVER or SPECURE-SERVER

Disable thinning: ☒ Default: false. Careful! Upload overhead.

Ping

Period: Run test every x minutes (0 = test deactivated). Valid options: 0, 5, 10, 20, 30, 60!

Count: number of pings

Spacing: seconds; Interval between two consecutive pings.

Server:

QualcommAT

Active: ☐ Careful! Upload overhead.

Sampling Period: miliseconds

SAVE CHANGES

Figure 3.5: Screenshot of the CMPT settings web interface.

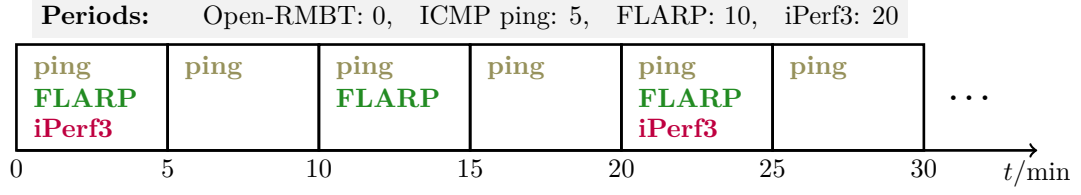


Figure 3.6: An example of different test periods. ICMP ping period is 5 min, FLARP period is 10 min and iPerf3 period is 20 min. This means that ICMP ping will be executed in every block, FLARP in every second and iPerf3 in every fourth. Open-RMBT test period was set to 0, which means disabled.

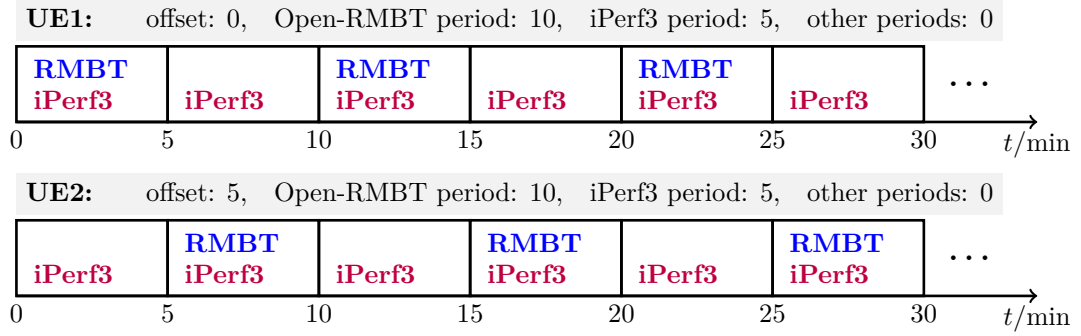


Figure 3.7: Example of two UEs with identical settings of test periods. The only difference is that the block offset of the UE1 was set to 0 min and the block offset of the UE2 was set to 5 min.

Field	Value
_id	"1509004310_357137070509830"
_rev	"1-b2cbece020fb2916e5e7749c15b01e69"
flarp	command "/data/data/at.tuwien.vraida.logger20/files/abeclient.bin -S 128.131.67.65 -P 2001 -c 0 -j 0" time_start 1509003916 response "Option: -S with argument: 128.131.67.65Option: -P with argument: 2001Option: -c with argument: 0Option: -j with argument: 0-+-+--+--+--+--+..." time_end 1509003916 results { }
iperf3	time_start 1509003946 time_end_UL 1509003957 time_start_DL 1509003957 time_end 1509003968 UL DL
logger	device_id "357137070509830" model "LG-K120" os "Android 5.1.1 LOLLIPOP_MR1; SDK 22" cpu_info "Processor : ARMv7 Processor rev 4 (v7l)processor : 0BogoMIPS : 2606.72Features : swp half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva..." battery location_coarse location_fine all_cells 0 timestamp 1509003881 cells 0 type "LTE" mRegistered true mTimeStamp 1292916299000000 identity_check "CellIdentityLte:{ mMcc=232 mMnc=3 mCi=4446466 mPci=52 mTac=14061}" identity Mcc 232 Mnc 3 Ci 4446466 Pci 52

Figure 3.8: A screenshot showing part of JSON document uploaded by CMPT application to the CouchDB database.

3.1.2 Web Interface

The web interface for changing CMPT configurations was implemented in php and is accessible via http://lisi4.nt.tuwien.ac.at/logger_settings/. After logging in, user can select any IMEI from the list of registered devices and change UE's configuration. Before applying any changes, the script checks and validates user's input and informs the user about the result.

Settings

The settings web page is shown in fig. 3.5. The first part (general settings) is well described in the screenshot, moreover we already talked about enabled/disabled randomization and link reset in paragraph "Scheduling." The applications of third parties (FLARP, iPerf3, Open-RMBT and ICMP ping) will be discussed later, QAT feature was already explained in paragraph "Passive Monitoring".

What remains is to explain the meaning of the test period which can be specified for every task and the block offset in the general settings.

The period of every task specifies how often this task is going to be executed. If the period of a task is 5 min, the task will be executed every 5 minutes, i.e. in every 300s block. If the period is 10 min, the task will be executed only in every second block, etc. The period 0 means, that the task is deactivated, it will be never executed. The period has to be nonnegative whole number divisible by 5. An example is shown in fig. 3.6 Note that the period only determines in which block the corresponding task is going to be executed, the location of the task slot within the block is still fixed or randomized, based on randomization settings.

The block offset can be used to shift the beginning of all blocks on the time axis by certain number of minutes to the right. This can be particularly useful in combination with proper settings of the test periods. An example is shown in fig. 3.7, where the Open-RMBT period is set to 10 min for two devices. By setting the block offset of UE1 to 0 min and the offset of UE2 to 5 min we make sure that the Open-RMBT tests will never overlap on the server side. Since the iPerf3 period is 5 min in our example, the different offset which is divisible by block duration will have no impact on iPerf3 test locations. It is also possible to set block offsets which are not divisible by 5 min, for this we have however not found any usage.

3.1.3 Server

Databases

Both, the settings database and the results database, are non relational⁶ document-oriented CouchDB [53] databases. Every entry is a JSON document with arbitrary structure. An example of one database entry (one JSON document) is shown in fig. 3.8. In order to extract data from the database one can use so called "views." We implemented them in JavaScript.

The document-oriented structure has the advantage, that we can easily add new tests and features to the CMPT Android app without need to do any modifications in the database – all collected data are simply stacked to single JSON document in the final slot and uploaded in the "UL results, DL config." slot to the database over HTTPS request.

Similarly, any new settings entries can be added without having to care about database structure, only the web interface and the CMPT app have to be modified. CMPT app downloads settings over HTTPS request in "UL results, DL config." slot.

⁶Also called NoSQL.

Automatic Configuration Changes

If the auto configuration option is activated in the settings (fig. 3.5, part General), then the script `auto_conf.php` which is located on the server is allowed to change configuration automatically. The script is executed by Cron every five minutes (i.e. once per block) and can perform arbitrary configuration changes. In the subsection “3.1.5 Examples of Use Cases” we explain how this can be useful.

3.1.4 Tools of Third Parties

The compiled binary files and the Android .apk files are stored on the server and their versions are stored in the settings database. If we want to replace some third party app with newer version, we just replace the file on the server and increase the version number in the settings database. The CMPT Android app then recognizes that version number has been changed and automatically downloads (and in case of .apk files also installs) the new file.

- **iPerf3:** Binary file. We already mentioned iPerf3 in the first chapter several times. The documentation can be found at [30]. In the settings the following can be changed: test duration, protocol selection (TCP or UDP), number of parallel TCP connections or UDP streams, target server and port. In the iPerf3 slot one DL test of given duration and one UL test of given duration is performed. Slot duration is 30 s, i.e. selected test duration must not be larger than 15 s.
- **Open-RMBT TU:** Android .apk file. The whole section 3.2 is dedicated to this compilation of Open-RMBT. We should take care that test duration is not set too high. RMBT slot is 120 s, but the Open-RMBT performs ping test, quality of service test⁷ (QoS), throughput DL and UL test. The test duration applies only to DL test and UL test (it is the same for both), the rest is fixed.
- **FLARP:** Binary file. Fast Lightweight Available Rate Probing, which was developed by Michael Rindler [54]. It is included because Michael used CMPT for his own measurements. In this thesis we don’t analyze FLARP results.
- **ICMP⁸ ping:** Invokes the Android’s (i.e. Linux) built-in ping shell command with count, spacing and target server specified in the configuration settings. Since this thesis focuses on TCP over IP throughput measurements, we don’t analyze ping results in following chapters.

3.1.5 Examples of Use Cases

In this subsection we demonstrate the versatility of the CMPT framework by mentioning several different scenarios in which it was applied.

Passive Monitoring

The first version of CMPT was used in MobCom Weather project to passively collect signal strength data of all reachable mobile cells. The signal strength data were then compared with precipitation measured by nearby weather stations to find out whether there is any correlation.

⁷Loading some web pages, etc.

⁸Internet Control Message Protocol

Randomized Sampling

The idea behind the randomization within every block is that we want to avoid periodic sampling when performing throughput tests like iPerf3 and RMBT. Randomized sampling has the advantage that it can capture different frequencies of the measured process.

In the end we have a trade off between random sampling and periodic scheduling – periodic scheduling assures that a test is performed in every k -th block, randomization assures that the position within each block is random. The position can't be completely random because we need some block structure in order to upload results at some point and e.g. also limit the number of performed tests to control the mobile data consumption.

Assuring no Interference

Sometimes we want to run tests on several different devices and we want to make sure that e.g. not more than one RMBT test is running at the same time in one mobile cell / against one RMBT server, etc. We can set appropriate test periods and block offsets similarly to fig. 3.7. In case of, for example, four UEs we would set test period to 20 min and block offsets to 0, 5, 10 and 15 for UE1, 2, 3 and 4.

Intended Interference

This is the opposite of the previous case. In one measurement scenario Michael wanted to compare test results without interference and test results with interference. Measurements were performed in reference cell of operator A1 in order to make sure, there are no other active users in the cell.

On the UE1 the test period was set to 5 min, on UE2 the test period was set to 10 min. Randomization was disabled, therefore in every odd block all tests on the UE1 and UE2 started at the same time. In every even block there were no tests running on the UE2.

Automatic Configuration Changes

In another scenario we wanted to compare how the throughput measurements are influenced by different number of parallel TCP connections. We measured in live LTE network. Because of the time-of-day effect⁹ we made sure that different configurations are tested close to each other (testing one configuration on Friday and another one on Saturday wouldn't be good idea, as the results will differ due the cell load).

Changing the configurations of several devices every five minutes manually wouldn't be very practical, therefore we used the script for automatic configuration changes to modify configuration after every 300s block. This method was also used in [55].

3.1.6 Outlook and Limitations

In the future we would like to make the CMPT framework open source. At this point the framework is quite generic but some changes like adding a new third party application can be achieved only by modifications of the source code. It shouldn't be too challenging to modify the CMPT app and the web interface in such a way, that user could define his/her own block length, specify sources of his/her own binary files or apk files, specify which tasks should be fixed and which should be randomized.

Currently there is just one script for automatic configuration changes, which can be only enabled or disabled for every UE. Modifications have to be done directly at the script file. It would be better if every user could define multiple scripts for any of his/her UEs.

⁹Between 2 and 3am the mobile cell is almost empty, whereas in the afternoon we share the cell with many other users. Moreover different days have different trends. We will see that in the next chapter.

The CMPT application is not optimized for usage in 2G networks and other scenarios where the maximum achievable data rate does not allow to finish the upload within the 30 s “UL results, DL conf.” slot. A hand on solution is to perform e.g. just passive monitoring in order to reduce the amount of data uploaded. Another solution would be to store results locally at the device (this however requires large amount of free storage at the UE and additional efforts, someone has to take the UE and copy results to the database manually). A viable solution to extend the CMPT framework such that user could configure the length of the “UL results, DL conf.” slot as needed, or to extend the CMPT framework in such a way that the device would perform measurements in e.g. 2G network and then connect to some faster link (if available; e.g. to WLAN or LTE) to upload the results.

3.2 RMBT

Austrian Regulatory Authority for Broadcasting and Telecommunications (RTR)¹⁰ provides an Android application RTR-NetTest which informs users about the current service quality (UL data rate, DL data rate, ping, signal strength) of their Internet connection [56].

The RTR-NetTest is available as open source under the name Open-RMBT [57]. RMBT stands for RTR Multithreaded Broadband Test. It was developed by alladin-IT GmbH and financed by the RTR.

The reason why we focus on RMBT—and the results generated by it—is, that we would like to use RTR’s Open Data – a publicly accessible database which contains measurement results of RTR-NetTest from many users at different locations. The RTR tool is itself a testing suite conducting several performance tests: TCP RTT measurement, UL and DL data rate measurement,... In the following we will provide a detailed overview of the procedure and the test phases.

3.2.1 Test procedure

A single RMBT test consists of seven phases. Detailed RMBT’s documentation is available at [58]. The phases 2–6 are marked in fig. 3.9.

Phase 1: Initialization

A connection between client and server is established. An encrypted tunnel is created in order to avoid any tempering in the communication.

Phase 2: Downlink Pretest

Duration of pretest is 2 s. Three TCP connections are established. They remain opened even after the pretest to be used for the actual DL test. The pretest ensures that the Internet connection is in an “active” state, e.g. HS-DSCH in UMTS [51] or DL-SCH in LTE [52], before the DL test. Pretest gives a rough estimate of the data rate. If the data rate is low, two connections will be closed and the test will continue with a single connection.

Phase 3: Latency Test

Tries to estimate TCP RTT. The client sends a short string to the server ten times in short intervals. Only one TCP connection is used, the other two (if not already closed at the end of the pretest) remain idle. The client measures the time between sending and

¹⁰Die Rundfunk und Telekom Regulierungs-GmbH.

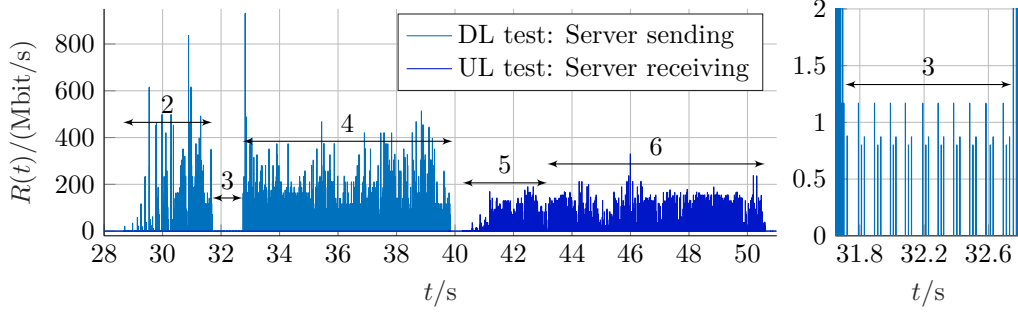


Figure 3.9: Wireshark's packet capture on the server side showing the phases 2–6 of the RMBT's test. The time on the x -axis is relative to the start of the packet capture.

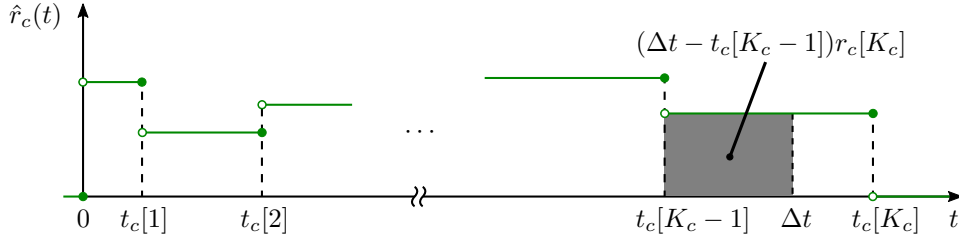


Figure 3.10: Average data rate of c -th connection in an interval $[0, \Delta t]$ is calculated as $\bar{R}_c = \int_0^{\Delta t} \hat{r}_c(t) dt$. It is equal to $(w_c[K_c - 1] + \text{c.t.})/\Delta t$, where c.t. stands for a correction term corresponding to the area marked with gray color.

receiving the return message, the server measures the time between sending its return message and the client's reception response. Note that this test typically differs from the ICMP ping used by, e.g. CMPT.

Phase 4: Downlink RMBT

Within each TCP connection the server continuously sends fixed-size data blocks of 4096 B. For every connection the client records K_c 2-tuples $(t_c[k], w_c[k])$, where $k \in \{1, \dots, K_c\}$. All transmissions start at the same time, which is denoted as relative time 0, i.e. $t_c[0] \triangleq 0$, $w_c[0] \triangleq 0$.

The default test duration is $\Delta t = 7$ s. After Δt the server stops sending on all connections. This means that $t_c[K_c]$ can be larger than the nominal test duration Δt because it takes some time until the data propagate through the network and arrive at the client.

The value taken as an approximation of the download rate for c -th connection is

$$\begin{aligned} \bar{R}_c &= \frac{1}{\Delta t} \int_0^{\Delta t} \hat{r}_c(t) dt = \\ &= \frac{1}{\Delta t} (w_c[K_c - 1] + (\Delta t - t_c[K_c - 1])r_c[K_c]) = \\ &= \frac{1}{\Delta t} \left(w_c[K_c - 1] + \frac{\Delta t - t_c[K_c - 1]}{t_c[K_c] - t_c[K_c - 1]} (w_c[K_c] - w_c[K_c - 1]) \right). \end{aligned} \quad (3.1)$$

This is illustrated in fig. 3.10. Note: we modified the notation used in [58] in order to be consistent with our notation established in chapter 2.

The approximation of the total download rate, for all connections together, is

$$\bar{R} = \sum_{c=1}^C \bar{R}_c. \quad (3.2)$$

Phase 5: Uplink Pretest

Analogous to phase 2 but with client sending for a duration of 2 s. The client opens three TCP connections to the server (if connections from phase 4 were already terminated; in other case the connections are reused). If the number of connections was reduced to one at the end of the phase 2, it will be reduced to one also at the end of this phase.

Phase 6: Uplink RMBT

Analogous to phase 4 but with client sending. Nominal test duration $\Delta t = 7$ s. The main difference is that the server has to report the measurements back to the client. The total download rate is calculated again according to eq. (3.1) and (3.2).

Phase 7: Finalization

The client sends the collected data to the server. . The report is stored in a centralized database containing all the details of the measurement, e.g. packet time stamps.

3.2.2 Thinning

When a user executes an RMBT test, the main result he/she is interested in is probably the total DL and UL data rate given by eq. (3.1), (3.2). To know more details, like the shape of the data rate curve, we need to work with as many (ideally all) samples $(t_c[k], w_c[k])$ as possible.

Upload of the test results from client to Open Data database consumes additional mobile data of the user. In order to reduce this overhead the RMBT performs compression before uploading the $t_c[k]$ and $w_c[k]$ samples.

This feature is not documented in [58], but we analyzed the source code [57], namely the files `RMBTClient.java` and `RMBTTest.java` located in the directory `RMBTClient/src/at/alladin/rmbt/client/`.

In the first file, in the method `runTest`, we find out that the total number of stored samples is limited to $K'_{c,\max} = \Delta t / \Delta_{\min} = 7 \text{ s} / 100 \text{ ms} = 70$ for every connection. Because $\Delta_{\min} = 100 \text{ ms}$, there will not be more than 10 samples per second for every connection, i.e. not more than 70 samples per second for the whole test.

The second file, in the method `addResult`, contains the compression algorithm itself. It is the thinning algorithm $\mathcal{T}: ((t_c[k], w_c[k]))_{k \in \{1, \dots, K_c\}} \rightarrow ((t'_c[k], w'_c[k]))_{k \in \{1, \dots, K'_c\}}$ described in chapter 2, section 2.3.

Command	Meaning	Default value
<code>--ez DISABLE_THINNING <bool></code>	See sec. 3.2.2.	false (thinning enabled)
<code>--ei NUMBER_OF_THREADS <int></code>	Number of TCP connection used for UL & DL test.	3
<code>--ei TEST_DURATION <int></code>	Duration of UL & DL test in seconds.	7
<code>--ei NUMBER_OF_PINGS <int></code>	How many times the client sends a string to the server in phase 3.	10

Table 3.2: Arguments which can be used when executing Open-RMBT TU.2.2.12.

3.2.3 More Control: Open-RMBT TU.2.2.12

In order to fully understand the properties of the system and to develop processing algorithms, we first focus on non-thinned sequences. As soon as we start to analyze Open Data results, we will have to deal with thinned sequences.

Open-RMBT TU.2.2.12 is a modified version of the client compiled by Leonhard Wimmer for internal usage of our working group. The main difference is, that when invoking Open-RMBT TU from Android’s command line, additional arguments (see tab. 3.2) can be specified. Open-RMBT TU has also the advantage, that it stores test results in JSON format to device memory, where CMPT can access it.

Furthermore we use our own RMBT measurement and database server¹¹ to make sure that there are not too many parallel tests running on the server side and also to have a possibility of controlling the server and retrieving packet captures when needed.

3.2.4 Open-RMBT Applications in Different Countries

We already mentioned RTR-NetTest and Open-RMBT TU. We have however discovered other tools which are just compilations of Open-RMBT with different appearance and slightly changed test configurations. This allows us to access even more open data from other countries.

Slovenian Agency for Communication Networks and Services (AKOS) provides AKOS Test Net [59], open data are available at [60]. Slovakian Regulatory Authority for Electronic Communications and Postal Services provides MobilTest operated by SPECURE GmbH [61], open data are available at [62]. Serbian Regulatory Agency for Electronic Communications and Postal Services (RATEL) provides RATEL NetTest [63], no open data were found.

In the table 3.3 we summarize list of tools which are all based on Open-RMBT.

App name and country	Test duration	TCP conns.	Version
RTR-NetTest (Austria)	7 s	3	RMBTws 0.8.0
AKOS Test Net (Slovenia)	5 s	3	RMBTws 0.3
MobilTest (Slovakia)	5 s	3	<i>not specified</i>
RATEL NetTest (Serbia)	5 s	3	RMBTws 0.3
TU-RMBT (TU Wien)	arbitrary	arbitrary	Open-RMBT TU.2.2.12

Table 3.3: An overview of different Open-RMBT compilations comparing duration and number of parallel TCP connections of DL and UL data rate test.

¹¹Many thanks to Leonhard Wimmer and Michael Rindler!

Chapter 4

Evaluation of Controlled Measurements

In chapter 1 we performed just a limited number of measurements in a manual setup not using CMPT framework to analyze the properties of the network under test. Now, after introducing the notation and theoretical concepts of chapter 2, we are finally ready to evaluate large measurement campaigns conducted with CMPT framework described in chapter 3.

4.1 Simplified Notation

In this chapter, wherever possible, we simplify notation of chapter 2 to achieve better readability. Lower index c still denotes separate connections, its absence means that we talk about merged quantity. Resampling period $T = 1\text{ ms}$, size of smoothing window $n = 101$.

Resampled rate is plotted in time-continuous representation and denoted $R(t)$. We use discrete representation $R[k]$ when talking about short time Fourier transformation and index shifts. This shouldn't cause any confusion, since there is always one to one mapping between time-discrete representation $R[k]$ and time-continuous representation $R(t)$.

Smoothed rate will be denoted as $S(t)$, even though it is discrete function we place the samples such that k -th sample corresponds to time $t = kT$. In cases where different size of smoothing window or different shape are used it will be explicitly stated, but we still use the same notation $S(t)$.

4.2 Presence of Data Rate Oscillations

In many RMBT's results we observed strong oscillations (fig. 4.1, second and third row) of the total resampled data rate R which are not supported by the basic theoretical assumptions made in chapter 1. Therefore we decided to collect packet captures on client side and server side and compare them with results reported by RMBT. We will see that oscillations can be suppressed by proper time shift of individual TCP connections.

Measurements were conducted in LTE network of operator A1, in Vienna, using UE model LG F60. In this section we couldn't use the CMPT framework yet, because we needed to collect packet captures and analyze every test individually.

In addition to smoothing with rectangular window, which can be interpreted as explained in subsec. 2.2.3, we use also quadratic window.¹ Quadratic window has no such

¹It corresponds to smoothing three times with rectangular window. The shape of quadratic window is similar to cosine window.

straightforward interpretation but it leads to visually clearer trends (fig. 4.1, third row).

4.2.1 Measurement Setup 1

In the first, lightweight scenario the packet captures were collected using tcpdump on a smart phone (LG F60) while running RMBT data rate test. This measurement reveals that oscillations which are present in RMBT's data rate (fig. 4.1, red) do not occur in Wireshark's captures (fig. 4.1, green). The results support the theory that in fact the RMBT recording of packet events is the source of the observed oscillations.

We identified the following method to remove the oscillations. Figure 4.2 depicts resampled rate $R_c(t)$ for every connection separately. We noticed that Wireshark's data rate bins (fig. 4.2, lower plot) look, despite slightly different shape,² very similar to those of RMBT (upper plot). The main difference is, that the relative positions of the three connections differ. If we shift connection 1 by 31 ms and connection 3 by 32 ms (middle plot) and then sum them up, the oscillations disappear also in the smoothed rate S (fig. 4.1, purple). The mean squared difference between RMBT's and Wireshark's data rate is reduced:³

Smoothing	MSD before shift	MSD after shift
no	4708.6 (Mbit/s) ²	3091.0 (Mbit/s) ²
1 · 101	22.4 (Mbit/s) ²	6.0 (Mbit/s) ²
3 · 101	6.8 (Mbit/s) ²	3.1 (Mbit/s) ²

This is of course no general result but only one specific test. It should however demonstrate that RMBT's oscillations are caused by some time offsets of individual connections.

4.2.2 Measurement Setup 2

To compare the data rate on the client side with the data rate on the server side and assure that both traces are time synchronized, we have to design more complex measurement setup (fig. 4.3).⁴ We use an additional switch to mirror traffic in both directions, incoming and outgoing from the server's perspective. Mirrored traffic is captured on the laptop by Wireshark. On the laptop we also run CLI version of Open-RMBT TU.2.2.12 and collect packet captures from the client's perspective. In order to connect the laptop to LTE network we used a smart phone (LG F60) which tethers an Internet connection via USB.

An example of Wireshark's packet captures is shown in fig. 4.4. We have to discuss two issues: First problem is that in DL direction on the client side (upper plot: client receiving, dark green) we observe that measured data rate is limited to ≈ 120 Mbit/s. The "cut-off" is caused by USB limiting the maximum throughput. This is visible for rate R with $T = 1$ ms. If we however calculate average data rate over longer period (smoothed rate S in lower plot) the limitation disappears because LTE network under test doesn't reach such high average data rates in intervals longer than few ms.

²Bins of RMBT are lower, ca 150 Mbit/s, and wider, $2-4T$. Wireshark's bins are narrow, $1-2T$, and higher. Different shape could be caused by relative time offset $< T$ which would lead to different bins when resampling with period T , recall fig. 2.8, lower plot. Another difference is that Wireshark's captures are collected on network interface or at the network card driver, whereas RMBT obtains the packets after they are transmitted from network card to CPU. The constant difference in the smoothed rate is caused probably by header overhead.

³Before calculating MSD between Wireshark's and RMBT's data rate, and also before plotting fig. 4.1, the Wireshark's trace is shifted to match the time axis of RMBT's trace, i.e. such time shift that correlation between data rate of RMBT and Wireshark is maximized.

⁴Many thanks to Michael Rindler for his help with the measurement on the server side.

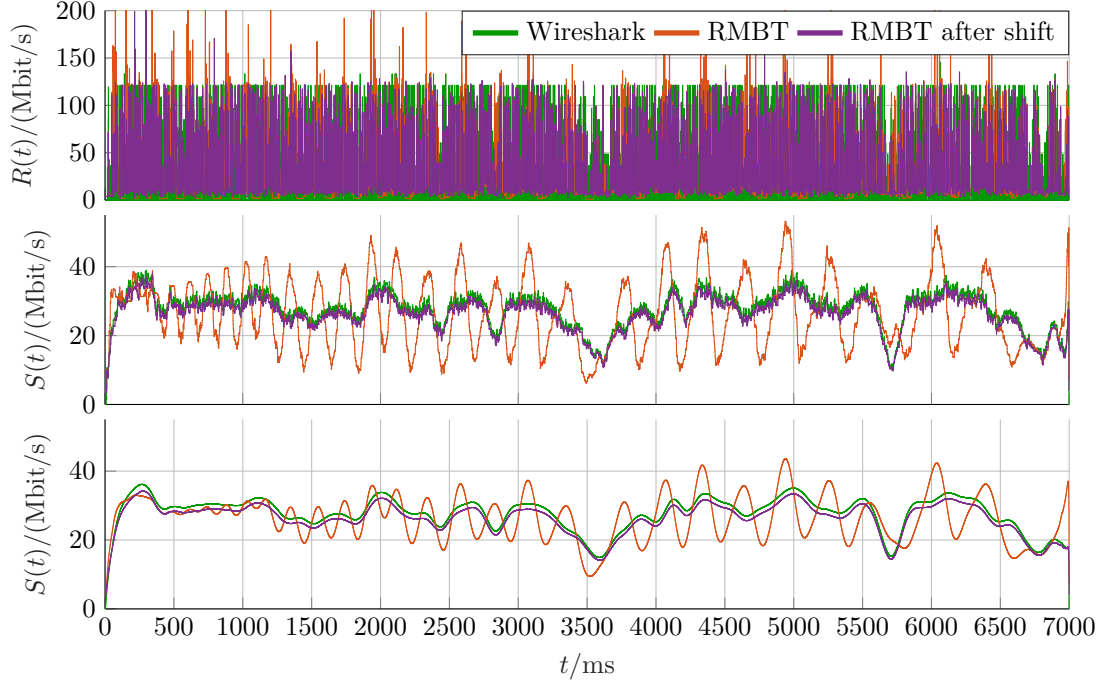


Figure 4.1: Comparison of rate reported by Wireshark (green) with rate reported by RMBT – before shift (red) and after shift of individual connections (purple). Top: Rate R with $T = 1$ ms. Middle: Smoothed with rectangular window of size 101 ($g_{101}[k]$). Bottom: Smoothed with quadratic window ($*g_{101}[k]$ three times).

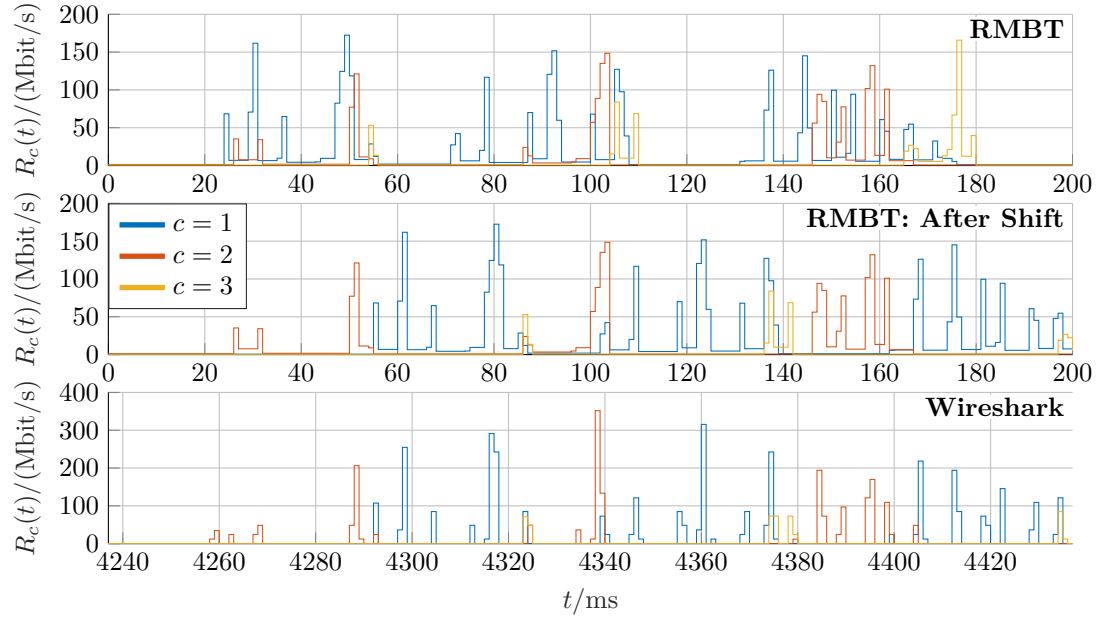


Figure 4.2: Detail of first 200 ms of RMBT's data rate test. Data rates are plotted for every connection separately. Upper plot: The resampled rates R_k of individual RMBT's connections. Middle plot: Connections 1 and 3 are shifted in order to match the Wireshark's data rates. Upper and middle plot: Time zero corresponds to beginning of the DL test as reported by RMBT. Lower plot: Time zero corresponds to the beginning of the packet capture.

Second problem is that during DL test the client receives more data than server sends. This can be recognized from fig. 4.4, where the trace “DL test: Client receiving” (dark green) drops to zero ca 0.5s later than the trace “DL test: Server sending” (lighter blue) and obviously leads to larger area below the curve. Also CDV plot in fig. 4.5 confirms this. The difference is probably caused by packet fragmentation – higher header overhead in receiver compared to sender. Although we found TCP segments of maximum length 23168 in captures at the server and segments of maximum length 1448 at the client, we have to be careful with the interpretation due to segmentation offloading at network interface card (NIC) [64], [65] (and very nice blog post [66]). To get more details we would have to collect captures at the link out of the hosts, for which we would need additional hardware.

Despite these two problems the comparison between what Wireshark captures and what RMBT reports clearly shows that the data rate trend during both – UL & DL test – is very similar, with the main difference that RMBT rates contain strong oscillations. Here we choose a test with stronger oscillations in order to show that smoothing with quadratic window does not help in such case (fig. 4.6 and 4.7).

This confirms observations from the first measurement setup: Data rate function reconstructed from CDV samples collected by RMBT contains in some cases strong oscillations which are not present in data rate reconstructed from Wireshark’s captures.

Also in the example presented in figures 4.4–4.7 the oscillations can be removed by appropriate time shift of individual connections, similarly to fig. 4.1, in this case connection 1 by 65 ms. MSD between Wireshark’s data rate and RMBT’s data rate is significantly improved:

Smoothing	MSD before shift	MSD after shift
no	1628.6 (Mbit/s) ²	500.8 (Mbit/s) ²
1 · 101	91.1 (Mbit/s) ²	2.3 (Mbit/s) ²
3 · 101	32.7 (Mbit/s) ²	2.0 (Mbit/s) ²

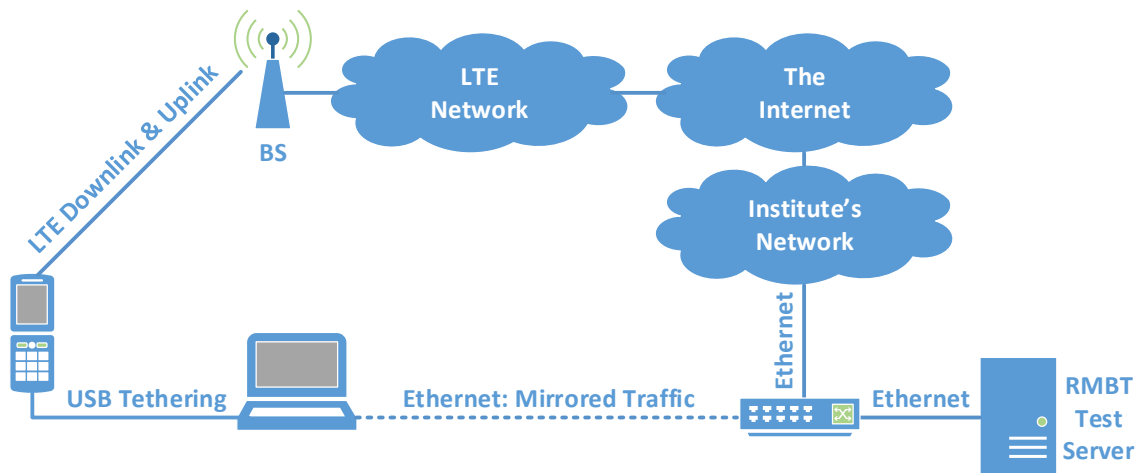


Figure 4.3: Measurement setup 2. LTE UE tethers an Internet connection to the laptop on which we run RMBT test and capture packets on the USB interface. The laptop also captures packets, which are mirrored from the server by a switch, on the Ethernet interface. Solid lines show the path the packets have to travel during the data rate test. Connection drawn as a dashed line is used just to capture packets also on the server side.

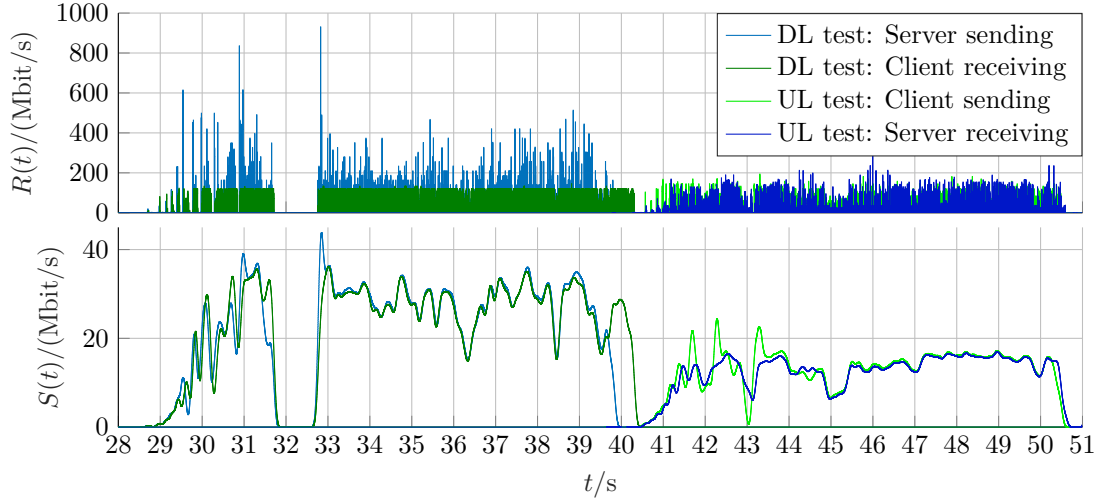


Figure 4.4: Wireshark’s captures on the client (green) and server side (blue). Upper plot: R , $T_s = 1$ ms. Lower plot: S , quadratic window (3·span 101; used to clearly visualize trends despite zoomed-out x -axis). In upper representation we can clearly see time boundaries of DL pretest, latency test, DL test, etc. Smoothing introduces certain broadening, on the other hand it clearly shows the trend – we can thus recognize that mean data rate during DL was ≈ 30 Mbits. Times on x -axis are relative with respect to the capture beginning.

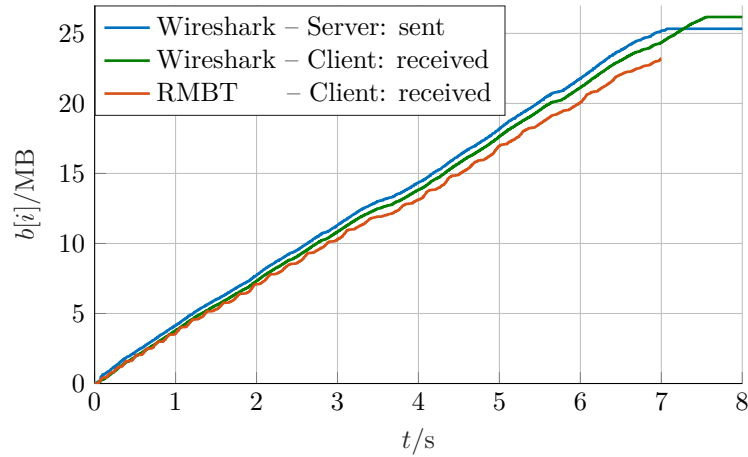


Figure 4.5: CDV vs time during RMBT DL test. We see that client receives more data than server sends. Samples reported by RMBT end around $t = 7$ s and already CDV trace indicates some oscillations. The different slopes of RMBT’s and Wireshark’s CDV are caused by header overhead.

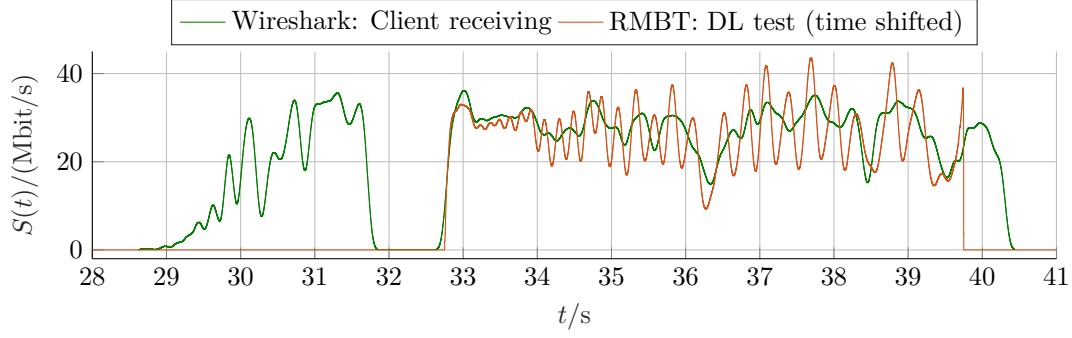


Figure 4.6: Comparison of RMBT’s DL test. Red: S (quadratic window) based on CDV samples reported by RMBT. Green: S based on Wireshark’s capture. The RMBT trace is shifted in time in order to match the DL test captured by Wireshark. The Wireshark trace contains also DL pretest.

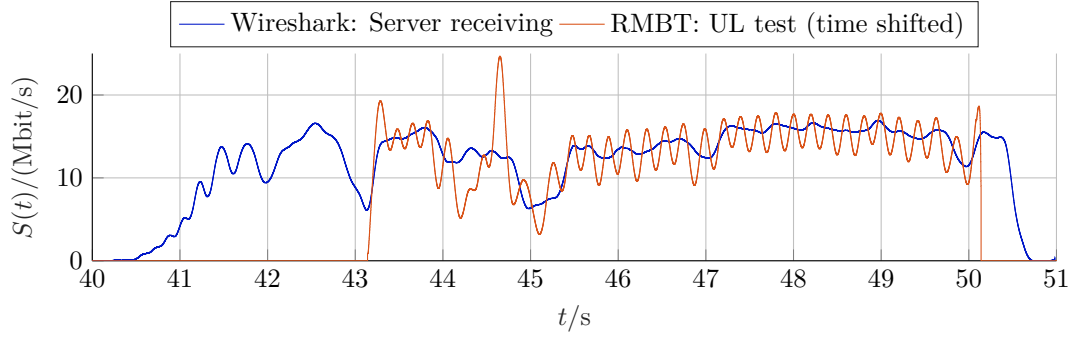


Figure 4.7: Comparison of RMBT’s UL test. Red: S (quadratic window) based on CDV samples reported by RMBT. Blue: S based on Wireshark’s capture. Here we compare RMBT with what the server is receiving, because RMBT’s samples are collected on the server side during UL test and then transmitted back to client, see phase 6 in subsec. 3.2.1. RMBT trace is shifted in order to match the UL test captured by Wireshark. The Wireshark trace contains also UL pretest and results upload.

4.2.3 Conclusion and Possible Cause of Offset

We saw that RMBT data rate tests show strong oscillations which are not present in Wireshark captures. By finding a proper shift for every TCP connection we can remove RMBT’s oscillations and obtain curves which are very similar to Wireshark’s captures.

We suspect that the time offsets of individual connections are caused by the RMBT client. As explained in sec. 3.2.1, the RMBT client (or server in UL) records pairs $(w_c[k], t_c[k])_{k \in \{1, \dots, K_c\}}$ for every connection. The time locations and cumulative volumes are measured relative to $w_c[0] = 0, t_c[0] = 0$ which correspond to the beginning of the DL or UL test. Our suspicion is that actually $t_1[0] \neq t_2[0] \neq t_3[0]$, due to scheduling of operating system or NIC. In the source code of the RMBT client the connections are programmed to start to transmit at the same time point but in reality there must be some switching between the connections – first connection transmits certain amount of data, then second connection transmits, then third and again first, second, third etc.

The switching between connections can be recognized in the shape of the data rate of individual connections of a multiple-connection test (fig. 4.8): Data are transmitted only in certain intervals on connection $c = 1$, followed by gaps where data are transmitted on the other connections. The shape shown in fig. 4.8 is similar also for the remaining

connections. With different number of connections (higher than one) the periods change but the principle is still the same and the alternating behavior remains.

For comparison, fig. 4.9 displays typical data rate trend of a single-connection test. It is not completely flat, there are notches followed by overshoots (probably larger number of segments passed to application after filling in a gap caused by lost or out-of-order packet), but we do not observe such regular, wide oscillations as in case of multiple-connection test.

In case of multiple-connection test the oscillations disappear after merging with proper offsets due to “destructive interference” – peak of one connection falls to notch of other connections. With wrong initial offsets we can get “constructive interference” instead.

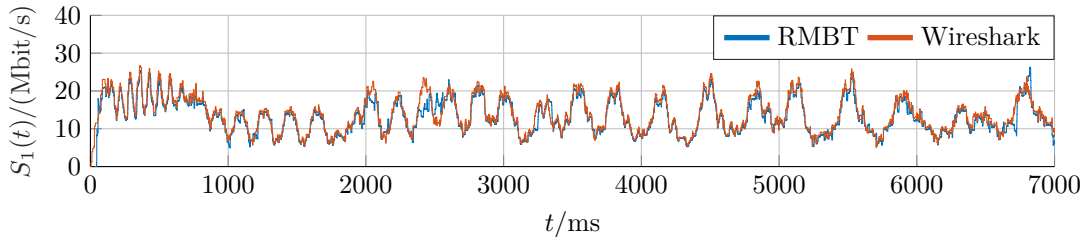


Figure 4.8: Smoothed rate on connection $c = 1$ in an RMBT DL test consisting of $C = 3$ connections.

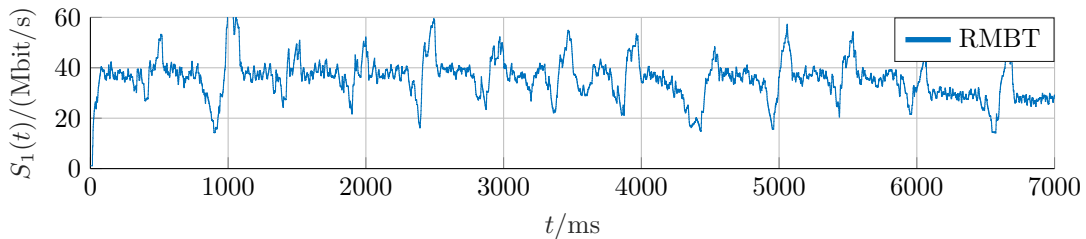


Figure 4.9: Smoothed rate of an RMBT DL test consisting of only one connection.

4.3 Systematic Removing of Oscillations

4.3.1 Automatized Measurements

To perform numerical evaluation we need to consider more than single tests in the previous section. For this purpose we implemented simple Android app (not the CMPT yet) which repeatably executes RMBT test with three TCP connections and duration of seven seconds (same setup as for results in RTR’s open data) and at the same time collects Android’s traffic statistics [67] to provide us an oscillation-free reference.

Traffic statistics are collected every ≈ 10 ms (time interval is not precise) and they consist of timestamp in milliseconds, number of bytes received and number of bytes transmitted (cumulative data volumes). The collecting starts when the RMBT is executed and stops after RMBT finishes. We have to detect the actual DL and UL test in these traffic statistics because we don’t know the exact beginning of the RMBT’s DL and UL test because of different RMBT’s phases. The detection algorithm marks all samples which correspond to rate lower than threshold (we picked 0.01 Mbit/s) and then identifies the beginning of the DL / UL test as the longest interval which does not contain more than 10 consecutive marked samples.

In this way we obtained around 50 tests with oscillation-free references. Reference signals and RMBT signals were resampled to $T = 1$ ms. Next we identify the time offsets for

RMBT's connections which minimize mean squared difference between RMBT's merged rate and the reference, eq. 2.26. Then we implement an approach without reference, eq. 2.25.

Despite the automatized measurement approach we still have to manually check the results of the detection algorithm and exclude wrongly estimated reference signals to make sure that we do not compare RMBT's test to traffic statistics corresponding to different test phase (e.g. pretest or result upload).

4.3.2 Minimizing MSD with Respect to Reference Signal

Here we solve the optimization problem in eq. 2.26 by exhaustive search. We are looking for $\hat{\kappa} = (0, \hat{\kappa}_2, \hat{\kappa}_3)$ which minimizes the MSD between RMBT's rate obtained by merging shifted rates of individual connections ${}^{\kappa}R[k]$ and the reference signal $R_{\text{ref}}[k]$.

An example of objective function is shown in fig. 4.12.⁵ For every pair of connection shifts $(\kappa_2, \kappa_3) \in \{-100, \dots, 100\}^2$ we calculate the MSD to reference signal. A complication is that the MSD is very sensitive to relative offset λ between ${}^{\kappa}R[k]$ and $R_{\text{ref}}[k]$ (fig. 4.10). For every (κ_2, κ_3) we thus find the lowest possible MSD. The computational complexity is therefore $\mathcal{O}(n^3)$. Due to the detection algorithm described above the relative shift λ should be hopefully close to zero.

The objective function shows one clear global optimum. We also notice that κ_2 and κ_3 can be optimized independently in this case (vertical and horizontal line), decreasing the computational complexity order by one. The diagonal line corresponds to optimal relative shift between connection 2 and connections 3.

4.3.3 Suppressing Oscillations Without Knowing Reference

The ultimate goal is to get rid of oscillations without knowing the reference signal.

Variance of Non-Smoothed Signal

Here we aim to minimize variance of non-smoothed signal $R[k]$, eq. 2.25, assuming that the oscillation-free signal should have minimum variance. This assumption may not be valid (e.g. if the unknown reference signal contains some high frequency components) but we have not much more options left. An example of objective function is shown in fig. 4.13.

The objective function shows similar pattern compared to fig. 4.12 – we can recognize traces of the same horizontal, vertical and diagonal line. But since the function jumps between local minima and maxima even for shifts by one sample, there is no obvious simplification possible, our only chance is the exhaustive search for all possible (κ_2, κ_3) .

Power in Frequency Band 2–15 Hz

An alternative approach is minimization of the energy just in a certain frequency band – we take 2–15 Hz. This is motivated by the fact that the oscillations we want to suppress have relatively low frequency. Fig. 4.11 shows spectrogram of the merged rate $R[k]$ before the shift of individual connections and after the shift. The y -axis is zoomed only to frequencies 0–15 Hz. Because $T = 1$ ms, the maximum frequency in the spectrogram is 500 Hz, but at such zoom level we wouldn't see much. The higher frequencies are much weaker than what is shown.

The corresponding objective function is displayed in fig. 4.14. Similar result is to be expected for minimization of variance of low-pass filtered (i.e. smoothed) signal – removing

⁵We plot the objective functions multiplied with -1 here for better visibility of the global optimum. Minimizing of MSD thus corresponds to maximizing $-\text{MSD}$.

all frequencies above 15 Hz and calculating power above 2 Hz should approximately equal power of non-DC components, i.e. variance, of low-pass filtered signal.

Note that also for this objective functions we see some traces similar to the horizontal, vertical and diagonal line as in case of the first objective function in fig. 4.12.

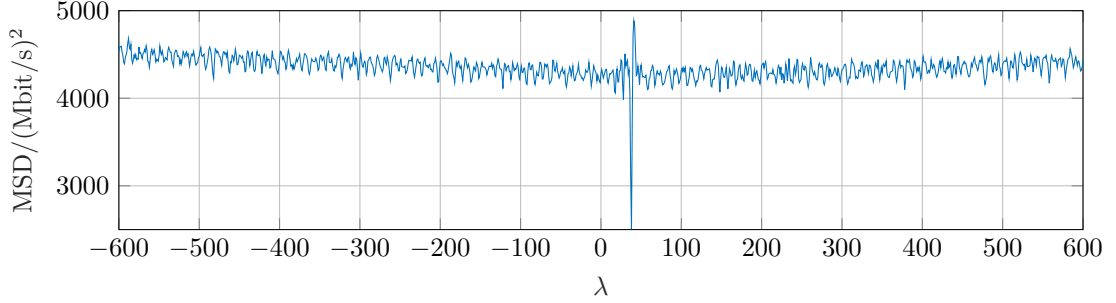


Figure 4.10: An example of mean squared difference between $R[k - \lambda]$ and $R_{\text{ref}}[k]$ as a function of shift λ , which shows that MSD is very sensitive to relative offset between the merged rate and the reference.

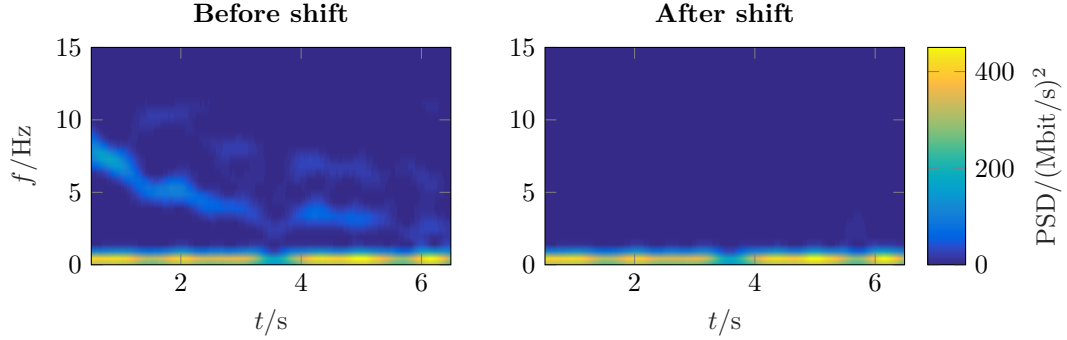


Figure 4.11: Left: Spectrogram of the merged rate $R[k]$. Right: Spectrogram of $\hat{\kappa}R[k]$ with $\hat{\kappa}$ minimizing the MSD between $\hat{\kappa}R[k]$ and reference $R_{\text{ref}}[k]$.

4.3.4 Numerical Evaluation

In this subsection we compare the resulting mean squared difference when optimizing different objective functions. This comparison is done only for DL, where the automatic test location detection operates reliable due to the presence of a larger gap between the pretest and the actual test (ping phase, recall fig. 3.9).

The rows of table 4.1 show: MSD – average mean squared difference (among all tests) between the given signal and reference signal; Δ_{MSD} – average MSD after subtracting minimum MSD; $\Delta_{\text{rel,MSD}}$ – relative difference between MSD and MSD_{min} .

$$\Delta_{\text{MSD}} = (\text{MSD} - \text{MSD}_{\text{min}}),$$

$$\Delta_{\text{rel,MSD}} = (\text{MSD} - \text{MSD}_{\text{min}})/\text{MSD}_{\text{min}} \cdot 100 \, \%.$$

MSD_{min} is obtained when minimizing MSD w.r. to reference signal, in our case the average $\text{MSD}_{\text{min}} = 767, 51 (\text{Mbit/s})^2$ (first row, second column). In the first row we obtain such high numbers because we considered non-smoothed signals, the reference has lower granularity than RMBT results (collecting traffic statistics with lower period than 10 ms was already causing lagging of our simple measurement app) and Android’s traffic statistics report values at the network layer, i.e. including TCP headers.

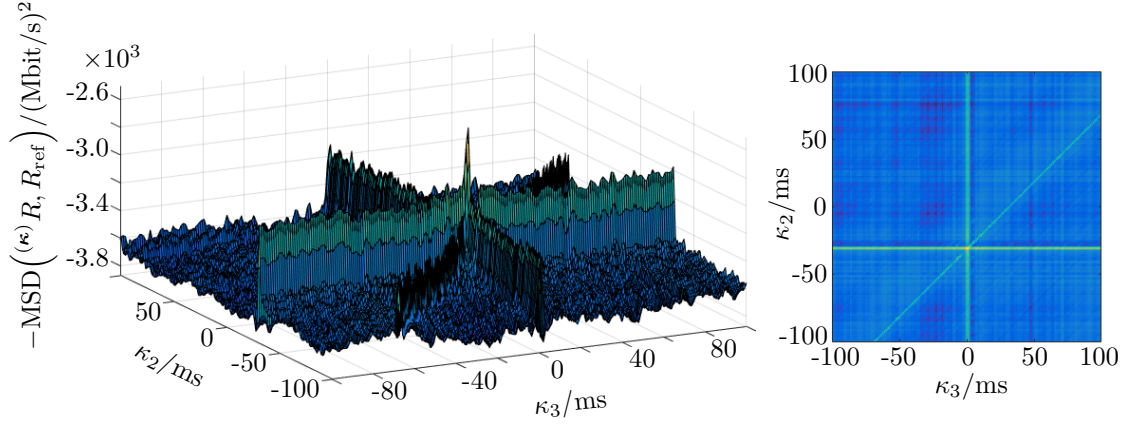


Figure 4.12: Mean squared difference w.r. to reference as a function of connection-shifts κ_2 and κ_3 . The global optimum is $\kappa_2 = -31$, $\kappa_3 = 1$, corresponding to -31 ms and 1 ms in time-continuous representation.

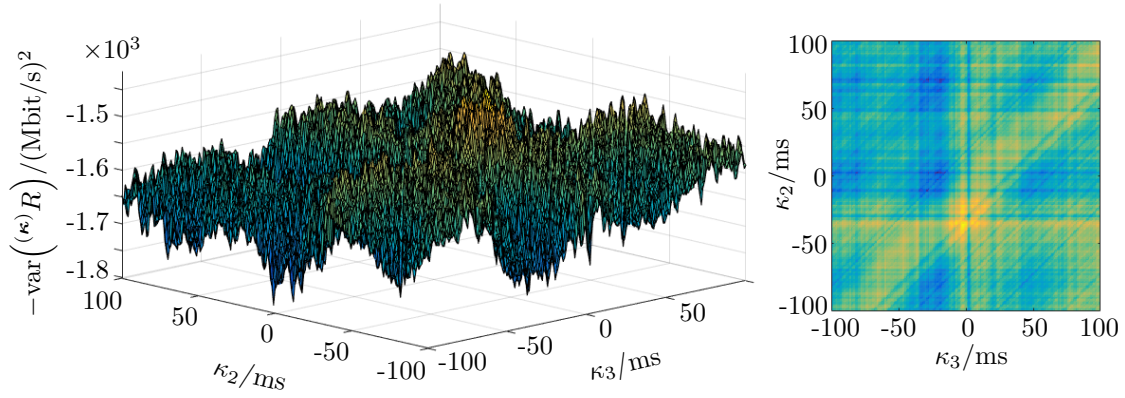


Figure 4.13: Variance of the resampled rate $R[k]$ as a function of shifts of 2nd and 3rd connection. The global optimum is $\kappa_2 = -36$, $\kappa_3 = -2$.

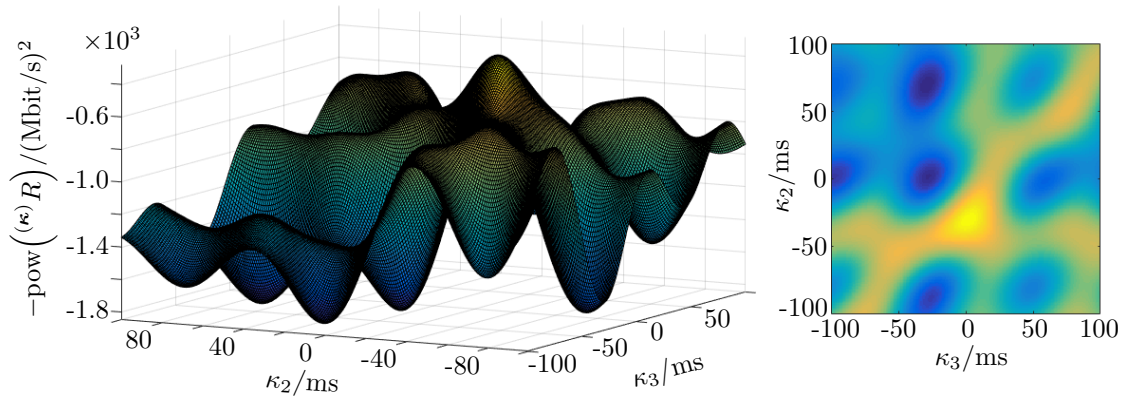


Figure 4.14: Total power in band 2–15 Hz as a function of connections shifts. The global optimum is $\kappa_2 = -31$, $\kappa_3 = 2$.

The second and third row show performance decrease when considering different signal than the one minimizing MSD. If we don't shift the connections at all, the average MSD increase is $105.77(\text{Mbit/s})^2$ (corresponding to 13,75 % difference). When we minimize total power in band 2–15 Hz, average MSD is increased by $66.25(\text{Mbit/s})^2$ (8.63 %). And when minimizing variance, MSD is worsened by $20.07(\text{Mbit/s})^2$ (2.62 %).

This supports the idea to minimize the variance in case of an unknown reference, as it results in lower MSD increase than when doing nothing or minimizing power in band 2–15 Hz. However compared to variance, the power objective function has the advantage that it is smooth. With grid search techniques we could decrease the number of tried shift combinations, therefore decrease the computational time in exchange for $\approx 6\%$ worse performance.

	before shift	minimizing MSD to ref.	minimizing pow. 2–15 Hz	minimizing variance
$\text{MSD}/(\text{Mbit/s})^2$	873.28	767.51	833.76	787.58
$\Delta_{\text{MSD}}/(\text{Mbit/s})^2$	105.77	0.00	66.25	20.07
$\Delta_{\text{rel,MSD}}$	13.78	—	8.63	2.62

Table 4.1: Performance comparison of MSD, variance of non-smoothed rate $R[k]$ and power in the frequency band 2–15 Hz.

4.4 Oscillations and Thinning Combined

Now we tackle both limitations of RTR's open data together: oscillations and thinning. Fig. 4.15 shows resampled and smoothed rates of all three connections after the proper shift, ${}^{\hat{\kappa}_c}S_c[k]$, and thinned rates obtained after the same shift, ${}^{\hat{\kappa}_c}R'_c[k]$. In first three rows we do not plot non-shifted rates R'_c they have same shape as ${}^{\kappa_c}R'_c$ and differ just in a time offset. As already mentioned in sec. 2.3.3, the thinning can be interpreted as noisy subsampling of $S_c[k]$ at given locations. As it can be seen in the first three plots, sometimes we hit a peak, sometimes local minimum but most of the times some value in between.

The problem is that the noisy subsampling at time locations with the spacing $> \Delta_{\min} = 100$ ms (which are moreover different for every connection) does not capture enough details of the smoothed rates $S_c[k]$ therefore even with proper shifts we don't observe complete destructive interference.

If we compare the total thinned rate with shift ${}^{(\hat{\kappa})}R'[k]$ (lower plot, magenta) to the total thinned rate without shifting $R'[k]$ (orange), there is no obvious improvement caused by the shift.

4.4.1 Discussion of Oscillations Model

The thinning obviously violates the Nyquist sampling theorem. Reconstruction of the smoothed rate out of the noisy samples would be possible only with certain model of S_c which would allow us to derive the shape of S_c , e.g. by LS fit.

In fig. 4.16 we again show smoothed rates of individual connections (gray) with thinned rates as noisy samples (orange), this time without the merged rate and with spectrograms of rates of individual connections. We attempted to build a simple model in sec. 2.4.1. The coefficients of exponential chirp were based on the red line in fig. 4.16 (fit of exponential function $\alpha + \beta \cdot \exp(-\gamma t)$ to strongest frequency components larger than 2 Hz).

The introduced model was not detailed enough to achieve complete removal of oscillations. Despite the model's simplicity we still obtained nonlinear model and nonconvex optimization problem with brute force complexity $\mathcal{O}(n^2)$. Interestingly we can find similarities in objective functions of the model and of the real tests. Fig. 2.12 has similar

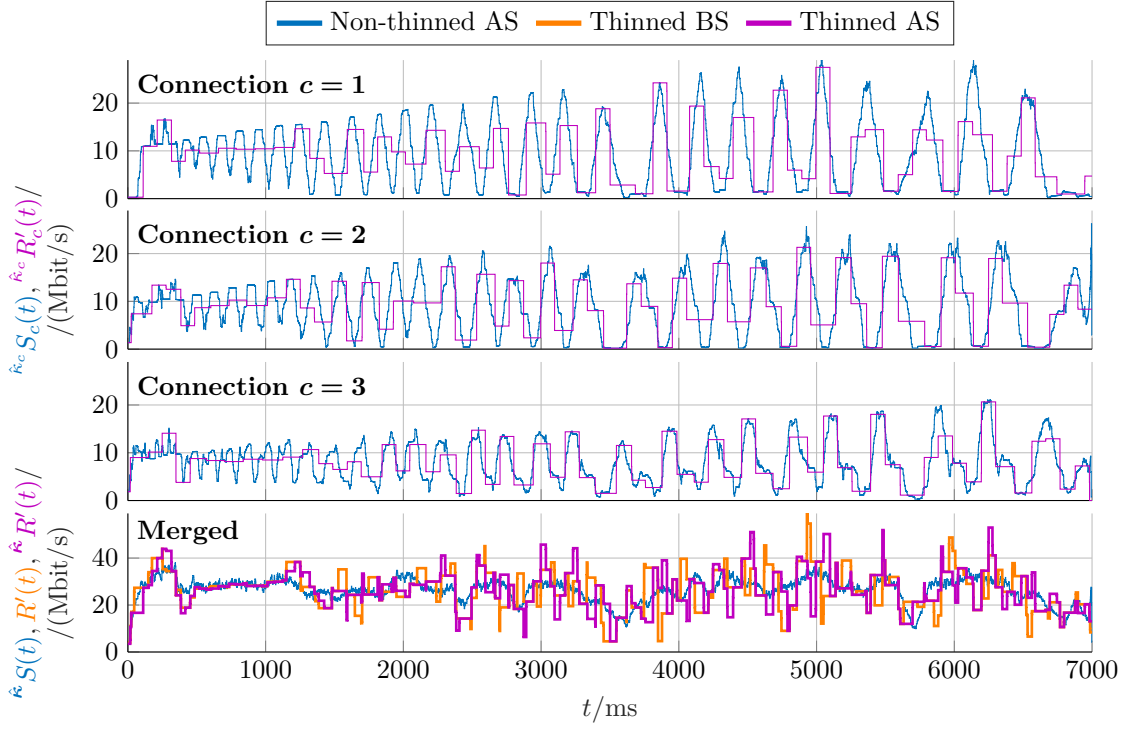


Figure 4.15: BS = before shift, AS = after shift. In first three rows we see resampled smoothed shifted rates $\kappa_c S_c$ and resampled thinned shifted rates $\kappa_c R_c$ of all three connections. The last row shows the total smoothed rate $\hat{\kappa} S$ after shift which removes oscillations. For the sum of thinned rates (magenta = after shift, orange = before shift) we don't see such improvement as in fig. 4.1 for the non-thinned rates.

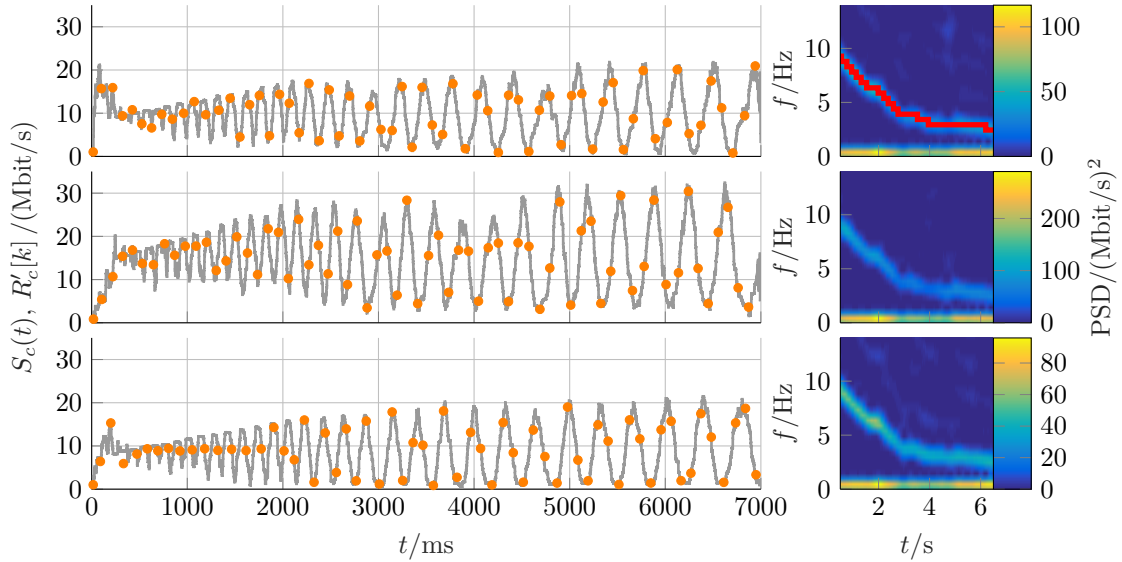


Figure 4.16: Smoothed rates S_c (gray) of individual connections, the corresponding spectrograms of $R_c[k]$ and thinned rates R'_c (orange) represented as noisy subsampling from S_c . The red line in the top spectrogram shows which data points we used for the exponential fit to construct the model in sec. 2.4.1.

vertical, horizontal and diagonal line as objective functions 4.12 (and to some extent also as fig. 4.13 and 4.14). In fig. 4.14 (and also in fig. 4.13) we observe regularly spaced local minima similar to objective function of phase shift 2.14 (with the difference that there we had opposite sign of the objective function).

4.4.2 Better Than Model...

Actually the best solution of the oscillations problem would be to figure out how to eliminate different time offsets directly at the RMBT client. The simplest solution would be to use just one TCP connection which is able to fill the BDP when window scaling is used.⁶

Regarding the thinning problem the best solution would be to report all samples using some clever compression technique to not increase the upload overhead much when reporting the results back to the server. Or, if the number of samples is supposed to be reduced, at least a resampling method proposed in sec. 2.1.2 could be used in order to have the same time grid for all connections.

4.5 Extension of Traffic Shaping Detection

In this section we provide an extension of token bucket traffic shaping detection from subsec. 2.5.2 using the concept of smoothed rate S from sec. 2.2.3.

Because the smoothed rate represents $n = 101$ different binnings with offsets mT , $m \in \{0, 1, \dots, (n-1)\}$, we obtain n different estimates of C , and in cases where level shift is successfully detected also estimates of ρ and σ . Fig. 4.17 shows an example of smoothed rate. Fig. 4.18 displays different estimates obtained for all 101 binnings.

The idea is following: Why should we prefer one binning and ignore all other offsets? Picking just one fixed bin position basically leads to randomly selecting one of many estimates. In fig. 4.18 in 25 cases no traffic shaping is detected. Using all estimates together gives us better opportunity to decide whether traffic shaping was present or not.

4.5.1 Repeated Measurements

We performed multiple measurements (151) with the same tariff. Using the idea of soft information processing, we don't hard-decide for a single estimate for every test but keep all estimates.

We tried different bin sizes (31, 51, 101 and 151 ms) and plotted the resulting histograms for estimates of C , ρ and σ (fig. 4.19). For C all bin sizes show similar distribution (strong peak at ≈ 10 Mbit/s corresponds to non-detected level shifts, in which case the C is estimated as an average rate of whole test). In case of ρ we obtained the most narrow distribution for bin size 101 ms.

Problematic is the estimation of σ where we obtain drift of distribution mode – larger bin size gives us larger mode. Problem is that smaller bin sizes can underestimate the bucket size due to earlier level shift detection caused by noisiness of the smoothed rate curve. Larger bin sizes can overestimate the bucket size.

4.5.2 Outlook

In previous paragraphs we briefly described the idea how to utilize the knowledge of multiple binnings with different offsets. The evaluation is still an unfinished task. We have to figure out which bin size does not underestimate / overestimate the bucket size – for this it would be good to know the ground truth, i.e. contact the ISP and ask them about

⁶This might have side effects. On some devices we saw that 1-connection RMBT tests are terminated earlier than after 7 s. It is difficult to say whether this is a vendor specific issue or a bug of RMBT client.

the implementation of their traffic shaping algorithm. Another possible modification is to soften the condition 1 in subsec. 2.5.1, as softer interpretation will allow to avoid detecting the level shift too early.

In the future we would like to evaluate also different approach: Use the smoothed rate to detect level shift location jointly for all binning offsets, rather than for every binning separately.

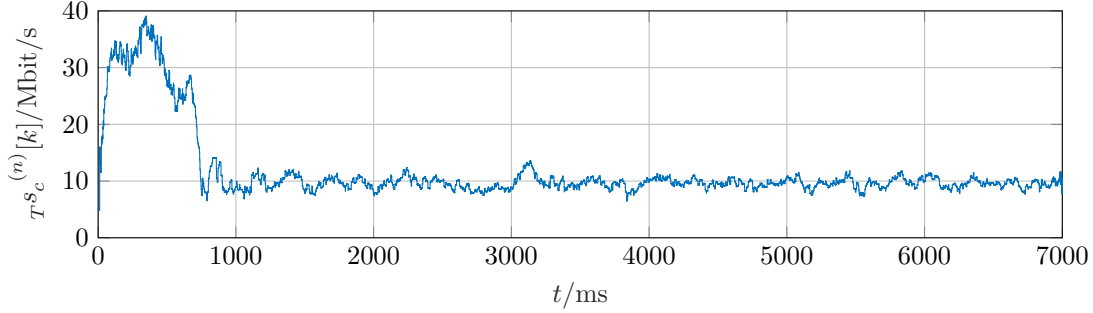


Figure 4.17: Example of smoothed rate S representing 101 different binnings with bin size of 101 ms in presence of token bucket traffic shaping. The different binnings are obtained by index mapping in theorem 5, subsection 2.2.3.

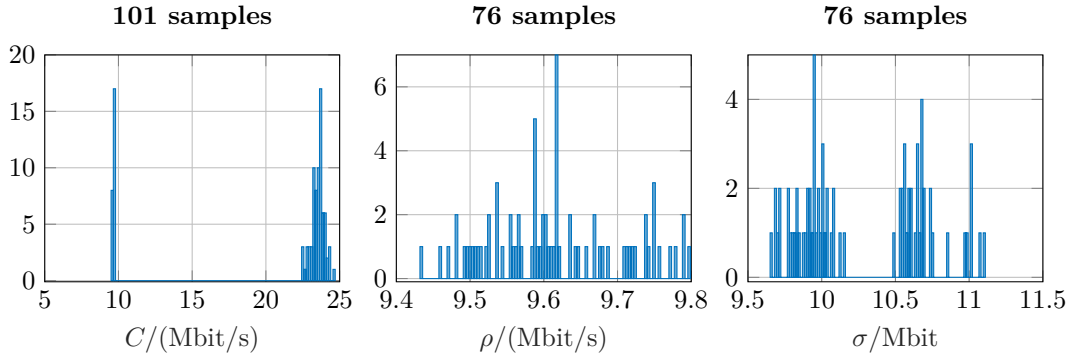


Figure 4.18: 101 estimates of peak rate C based on 101 different binnings of rate R (left). Level shift was detected in 76 cases – for these different estimates of shaping rate ρ (middle) and bucket size σ (right) are obtained.

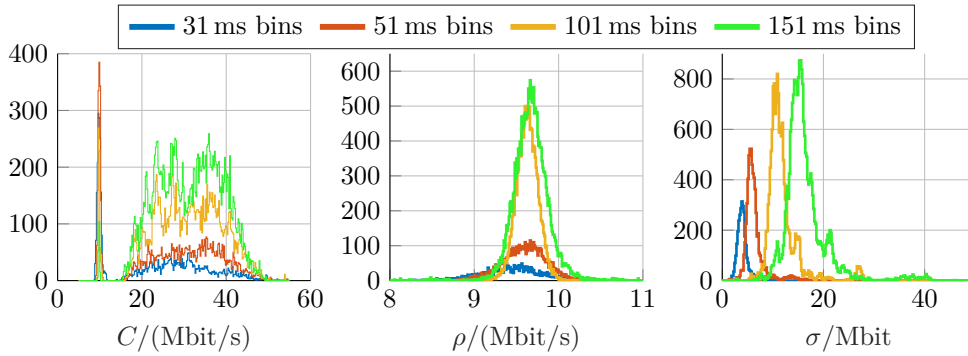


Figure 4.19: Histograms of estimates of 151 tests. For every bin size n we obtain n estimates per test, i.e. $151n$ estimates in total – this explains different heights of distributions.

4.6 Rate as a Function of Signal Strength and Time of Day

In order to allow for merging of data samples we analyze two main contributions. First, the relation between signal strength reported by the UE and the data rate reported by the test servers.⁷ Second, we analyze the impact of diurnal patterns, e.g. cell load, on the data rate measurements.

This section describes results of systematic CMPT measurements over longer time period. First we discuss situation in an unloaded reference cell in subsec. 4.6.1 where the measuring UE was the only active user and where the situation is easier. Then we go to measurements in live network in subsec. 4.6.2 and construct weekly trends of RSRP, RSRQ and DL and UL rate.

4.6.1 Reference Cell Measurements

Setup

Measurements were set up by M. Rindler who used them for analysis of his FLARP application. Since he used the CMPT framework we were able to schedule also RMBT and iPerf3 measurements and collect RSRP and RSRQ samples.

The setup was following: A shielding box contained the measuring UE and an antenna connected directly to base station over a cable with an attenuator. The link between the UE and the antenna was wireless because Michael then tested also other scenarios where multiple UEs were connected to the BS. In the scenario below there were no other devices connected to the BS. The attenuator was used to change the attenuation level at 2am every day.

iPerf3 and RMBT measurements were taken every ten minutes, i.e. in every second CMPT's block.

Results

Fig. 4.20 shows six days of measurements. The upper plot displays data rate measured by iPerf3 and RMBT in UL and DL (one point represents average data rate of the whole test). The middle plot displays RSRP in dBm and the lower plot RSRQ in dB (one point corresponds to five minute average – i.e. average RSRP / RSRQ in one CMPT's block).

The reason why we see fluctuations in RSRQ is the presence of noise (thermal noise + the shielding box can't fully attenuate interference from outside). RSRP stays more or less constant.

For the fifth attenuation level ($\text{RSRP} \approx -110.5 \text{ dBm}$) both, the DL and UL rate, are very stable during the whole day cycle. From this we can conclude that the daily and weekly trends we will see in next subsection are caused solely by situation in the RAN. The wired network links between the BS and test server are overprovisioned compared to the wireless link.

The larger variance of DL rate at other attenuation levels is caused probably by switching between different modulation schemes. We can't verify this directly because Android does not provide the information about measured SINR and calculated CQI. But if we have a look at average smoothed data rate curve at attenuation level 1 ($\text{RSRP} \approx -122.7 \text{ dBm}$) in fig. 4.21, we see two clear modes in DL – this could correspond to two different modulation schemes leading to different throughput. Similar bimodal distributions we observe also for other attenuation levels with exception of level 5 ($\text{RSRP} \approx -110.5 \text{ dBm}$), see fig. 4.22.

⁷Note that the signal strength reported by the device is coded as integer ASU (arbitrary strength unit) $\in \{0, \dots, 95\}$ which is mapped to $\text{RSRP/dBm} = \text{ASU} - 140$ and leads to 1 dB resolution [68].

With our UE of LTE category 4 it makes no sense to test lower attenuation than shown because in UL we already reach UE's maximum of 50 Mbit/s and in DL we closely approach the limit of 150 Mbit/s.

In tables 4.2 and 4.3 we list means and standard deviations of values plotted in fig. 4.20. The variance of data rate in case of reference cell is the minimum variance to be expected for certain signal strength level – in case of a real cell it only gets higher. We also conclude that there is no significant difference between RMBT's and iPerf3's mean data rate.

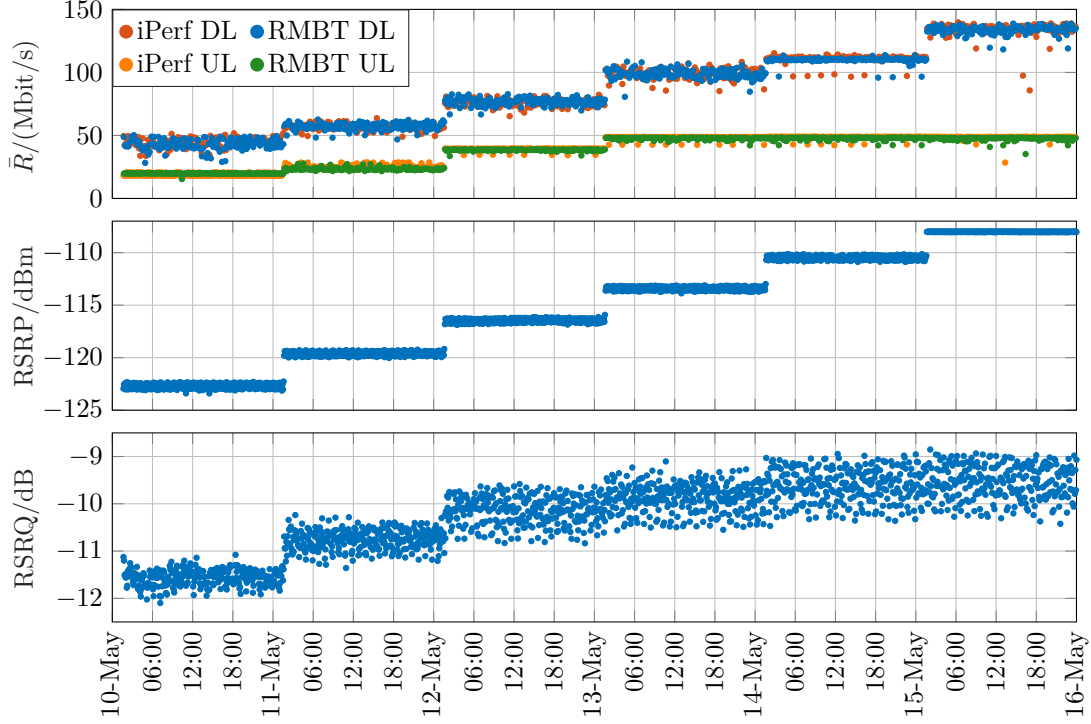


Figure 4.20: Measurements in reference cell between May 10, 2017 and May 16, 2017.

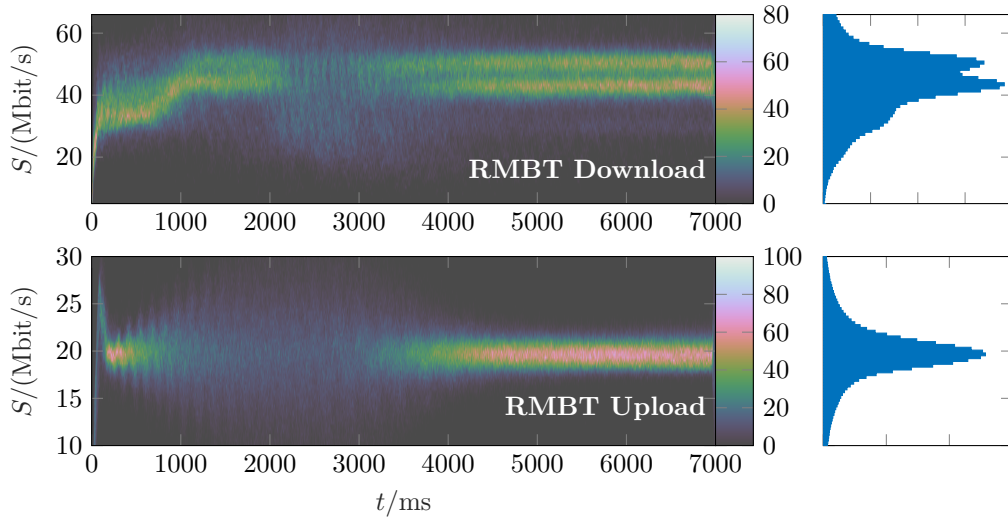


Figure 4.21: For every RMBT measurement we calculate the smoothed rate $S[k]$. The figure shows 2D histogram (left) of all tests at attenuation level 1 ($\text{RSRP} \approx -122.7$ dBm). The right subplots show histogram of all test samples. The two modes in DL indicate usage of two different modulation schemes.

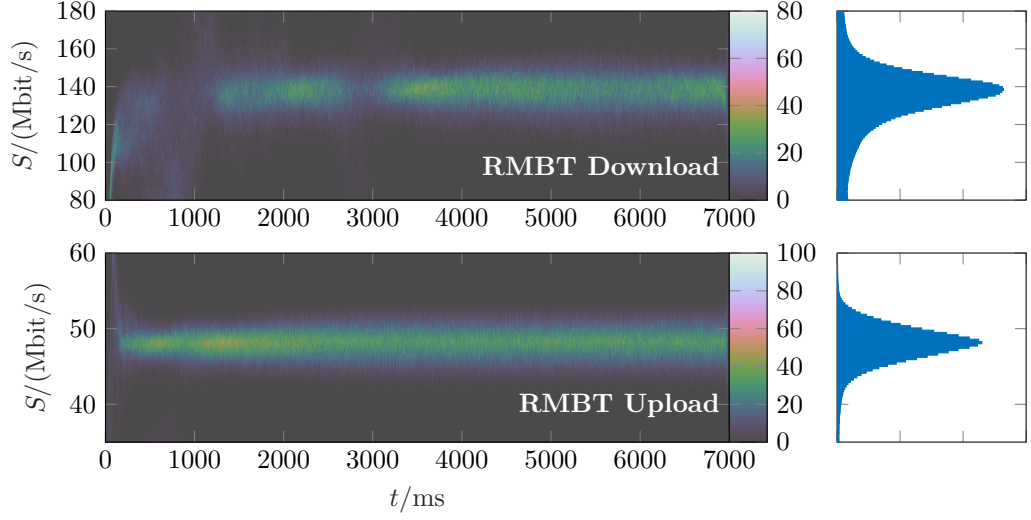


Figure 4.22: Averaged curves at attenuation level 5 ($\text{RSRP} \approx -110.5 \text{ dBm}$).

Attenuation level	RSRP		RSRQ	
	μ	σ	μ	σ
1	-122.70	0.26	-11.57	0.18
2	-119.61	0.22	-10.76	0.21
3	-116.47	0.20	-10.14	0.30
4	-113.42	0.20	-9.88	0.30
5	-110.49	0.20	-9.61	0.35
6	-108.02	0.02	-9.56	0.36

Table 4.2: Means and standard deviations (don't confuse with token bucket size σ) of RSRP (in dBm) and RSRQ (in dB) for the six attenuation levels shown in fig. 4.20.

Att. lvl.	RMBT DL		iPerf3 DL		RMBT UL		iPerf3 UL	
	μ	σ	μ	σ	μ	σ	μ	σ
1	42.88	4.26	42.23	3.61	18.79	1.11	19.77	0.38
2	56.95	2.88	56.90	2.38	25.48	1.24	23.46	1.09
3	76.67	2.98	76.01	2.95	39.13	1.13	38.42	0.70
4	98.88	3.82	98.84	4.03	48.45	1.26	47.52	1.12
5	110.21	0.98	111.59	0.99	—	—	—	—
6	133.39	3.60	134.12	6.02	—	—	—	—

Table 4.3: Means and standard deviations in Mbit/s for DL and UL mean rate \bar{R} of RMBT and iPerf3. In UL we give only values up to 4th attenuation level, at which the UE's limit is already reached.

Conclusion

In a reference cell in absence of other active UEs we obtain nearly constant data rates confirming that the bottleneck of whole path is the RAN, leading to the conclusion that also weekly and daily trends seen in next subsection are caused by cell load in the RAN.

The observed variations in data rate are caused with highest probability by BS's switching between two different modulation schemes, which we deduced from the two modes of averaged data rate curves. For average data rate at attenuation level 5 we detected only one mode leading to the standard deviation of ≈ 0.98 for RMBT and ≈ 0.99 for iPerf3 in DL (tab. 4.3, bold).

Note: RSRP and RSRQ means and std. deviations are already based on five-minute averages corresponding to CMPT's scheduling blocks.

Power of reference signal is determined by the attenuation level leading to stable RSRP. Presence of noise causes larger variations in RSRQ. Noise also causes variations in SINR and therefore possibly different CQI leading to different modulation and / or coding scheme used by BS. Android unfortunately doesn't provide SINR or CQI information at application level which makes whole analysis much more challenging, especially in a real cell occupied by other users.

4.6.2 Measurements in Live Network

Setup

Measurements were taken as a part of MobCom Weather project during September 8 – November 7, 2016 in Kindberg, Austria (rural area). UE running CMPT collected RSRP and RSRQ samples (every second) and executed iPerf3 and RMBT tests every hour.

Results

Fig. 4.23 shows one week of measurements. The behavior of RSRP, RSRQ and data rates is "wilder" than in case of reference cell (compare to fig. 4.20). Moreover occasionally a handover happens.⁸ Every RSRP, RSRQ point in fig. 4.23 represents average value in one CMPT's 5-minute block. I.e. if UE is registered in cell A for 40s and in cell B for 260s, the plotted point is an average of 40 samples from cell A and of 260 samples from cell B.

Since Android does not provide SINR, neither CQI, neither number of assigned resource blocks we can only take RSRQ as an indicator of changing cell load. This might work in cases where RSRP is stable, i.e. stable reference signal power, then RSRQ changes are caused only by changes of RSSI. More active users in cell increase RSSI, therefore we see decrease in RSRQ. Still we don't know modulation scheme and number of assigned REs, therefore we can't relate average rate and RSRQ directly.

In fig. 4.24–4.26 we see weekly trend of RSRP, RSRQ and of data rates. For RSRQ and RSRP we have one sample for every second, therefore we constructed smaller bins (1 hour) for the weekly trend. For data rates we needed larger bin size (3 hours) to reduce variations to some reasonable level. Vertical bars in all three plots show the standard deviation.

It is little bit naive to talk about distribution and standard deviation since we had only eight weeks of measurements (i.e. we have for every 1 hour or 3 hour bin only eight samples), but still we were able to show clear trend in cell load as a function of time of day and correlation between RSRQ and average data rate in case of stable RSRP.

The UE receives signal from multiple BSs, in fig. 4.24 and 4.25 we show two BSs for which the UE collected the highest number of samples. Despite the occasional handovers

⁸Another complication.

we conclude—based on the black RSRQ-line in the fig. 4.23 representing trend of PCI⁹ 16—that the UE was connected to the PCI 16 for most of the time (which is no surprise due to stronger RSRP and RSRQ). In RSRP plot we observe larger deviation between temporal 5-minute averages and weekly trend.

Conclusion

In this measurement scenario RSRP was quite stable therefore RSRQ changes are probably caused mainly by cell load changes leading to high correlation between RSRQ and DL / UL rate.

Such construction of weekly trends and such nice correlation between RSRQ and data rates is not always possible, especially in urban areas. We need a relatively stable RSRP level and similar cell load behavior for several weeks (i.e. no holidays, no summer/winter time shift, etc., which usually change users' behavior).

The whole problem is complicated moreover by the fact that UE selects different base stations (in this case two), e.g. due to changing cell load and interference caused by neighboring cells.

Outlook

Ideas for further work are following. First, subtract the weekly trend to allow fairer (ideally cell-load independent) comparison of measurements taken at different days and day-times. Second, model weekly trend to allow subtraction also in cases where we have not so many consecutive repeated measurements.

In case there are larger deviations of measurement values from the weekly trend (like in case of RSRQ in fig. 4.23) the trend subtraction can be used to reveal anomalies due to irregular events in the network.

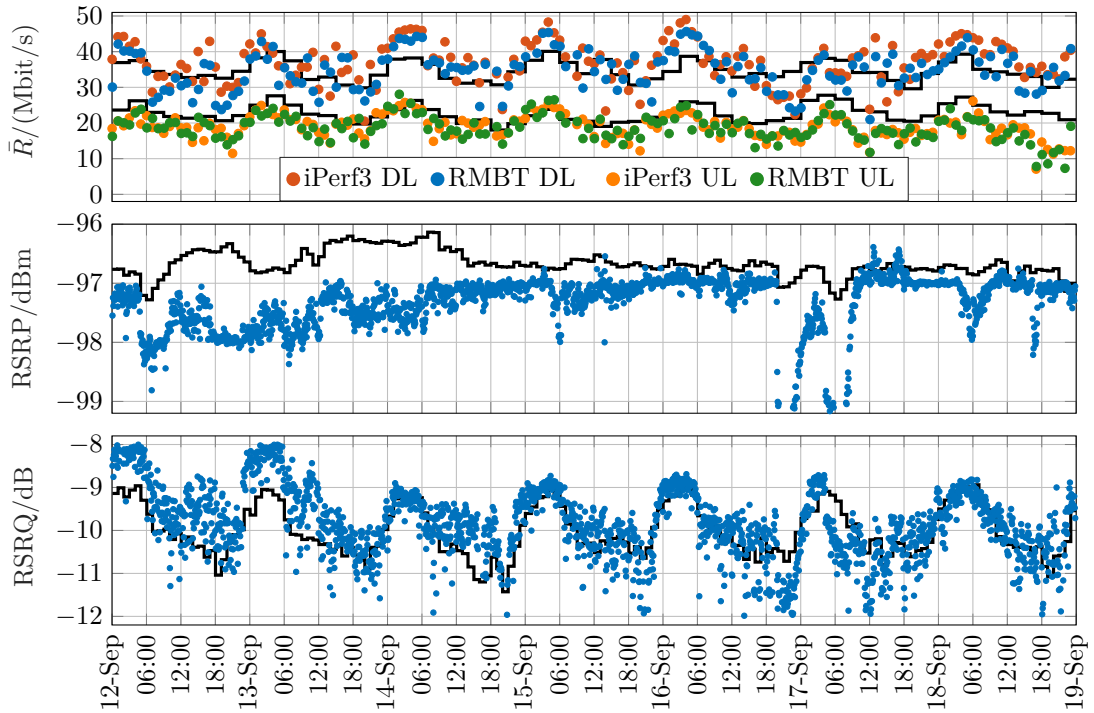


Figure 4.23: One week of measurements (September 12 – September 19, 2016) in Kindberg, Austria in a live LTE network. Black lines represent trends from figures 4.24–4.26.

⁹Physical Cell Identity. The trend was taken from fig. 4.25.

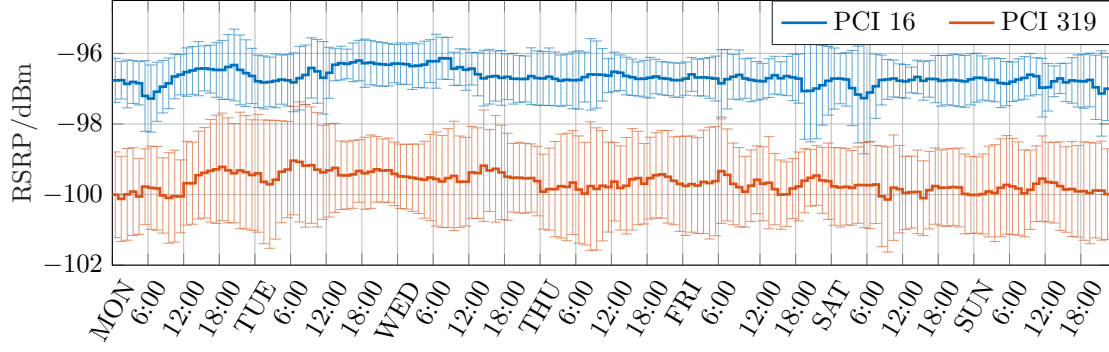


Figure 4.24: Weekly trend of RSRP for two nearest base stations.

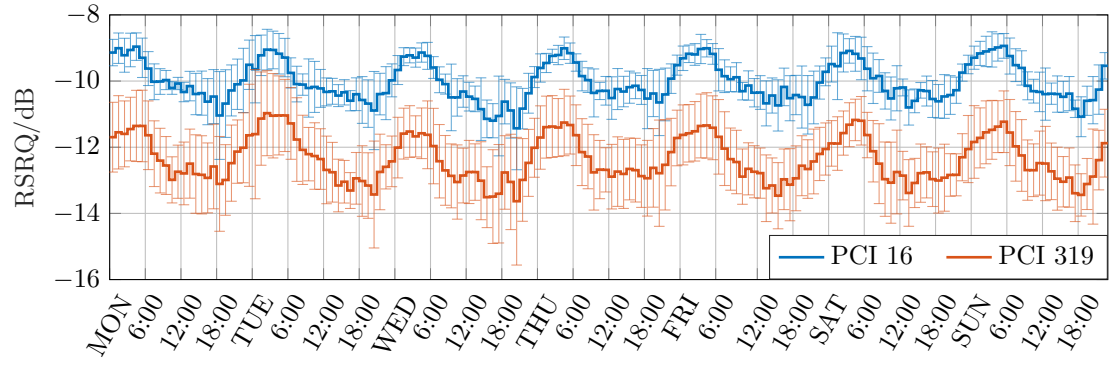


Figure 4.25: Weekly trend of RSRQ for two nearest base stations.

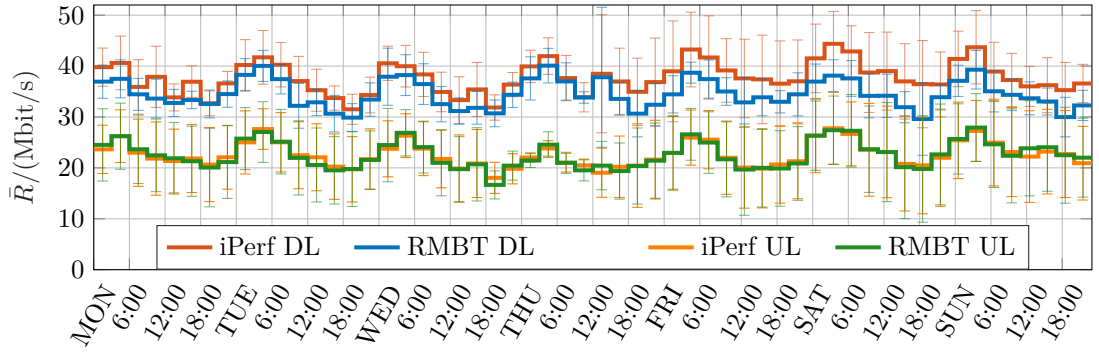


Figure 4.26: Weekly trend of UL and DL data rates measured by iPerf3 and RMBT.

4.7 Test Shortening

In [55] we discussed a possibility of shortening data rate tests without significantly impact the measured data rate. We performed only 30 s measurements and by cutting off last $30 - n$ seconds we were able to emulate shorter tests of duration n seconds. Measurement results have shown that there is a trade-off between test duration and measurement error.

We compared also different numbers of TCP connections, the results support the idea of merging performance tests with different configurations.

Room for improvement: First, to come up with a model describing the error as a function of test duration. Second, find more suitable reference with lower bias – in the paper the reference was part of the test. Bootstrapping can't be used because data rate is not stable in live network (recall fig. 1.7, red curve). For future work it might be interesting to derive reference from averaged curves (e.g. fig. 4.21) but only if we don't observe multiple modes.

Chapter 5

Crowdsourced Data for Network Performance Metrics

In the previous chapters we’ve seen that fixing impairments of open data and processing every test separately is very challenging and time expensive, requiring high computational complexity.

An alternative way is to come up with lightweight metrics which extract just few important parameters to characterize the whole test. There is no straightforward recipe how to accomplish such task, what we need are clever ideas. Up to this point we have come up with two lightweight metrics which are presented in sections 5.2 and 5.3. First we briefly describe structure of open data set in sec. 5.1.

5.1 Structure of Open Data

RTR’s open data provide not only thinned cumulative volume sequences of all test connections, but also GPS coordinates, UE model, network operator, network technology, signal strength (indicators depend network technology), etc. Exhaustive list is stated at the official documentation [38]. Not all parameters are always reported – user may e.g. decline to share GPS location, certain UEs don’t report their model,...

5.1.1 Passively Active Measurements

RTR-NetTest performs active measurements. The passivity rests in the fact that we have no control when and where which user starts a test. Daily trend in fig. 5.2 (taken from [69]) illustrates that most of the tests is started in the afternoon, the minimum of tests is started in early morning.

The depicted trend might be a good indicator of cell load during the day. It is inverse of the daily trend of RSRQ and data rate (fig. 4.25, 4.26). I.e. open data are biased in the sense that most of the tests is conducted at high cell load leading to lower average rate.

5.1.2 Feature Filtering

Fig. 5.1 depicts all LTE tests in Vienna between August 6–17, 2017 (i.e. 12 days). There were 1547 of them. If we further filter for a single operator, select smaller time span (e.g. 24 hours) or more specific location, we run out of samples very quickly, even if we allow all UE models, all signal strength levels, etc.

5.1.3 Outlook: Possible Solutions

A solution for diurnal patterns could be a model of the daily trend which would be subtracted from the data rate as already mentioned. For creating more spatial samples, an interpolation using Gaussian processes could be used. This is however beyond the scope of this thesis.



Figure 5.1: RTR-NetTest spatial distribution example. Source: internal software tool of Mobile Comm. group, Institute of Telecommunications, TU Wien.

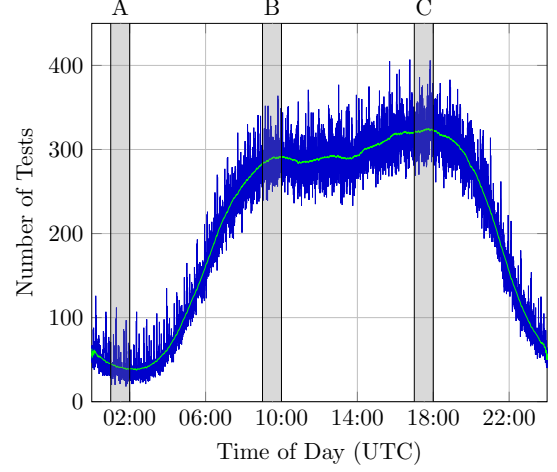


Figure 5.2: Number of RTR-NetTest tests started (aggregate 2014–2015) as a function of time of day. Source: [69].

5.2 Tariff Limitation

The first metric is called peak to average ratio (PAR), which we define as follows:

Definition 5.2.1 (Peak to average ratio).

$$\text{PAR} \triangleq \hat{R}/\bar{R},$$

where \hat{R} is the maximum 100 ms average rate in the first two seconds of the test:

$$\hat{R} \triangleq \max_{k \in \{n-L, \dots, 20n-L\}} \left\{ T^s(n)[k] \right\} \quad \text{with } n = 100 \text{ and } T = 1 \text{ ms},^1 \quad (5.1)$$

and \bar{R} is the average rate of the whole test as defined in 3.2.

In case of open data only thinned sequences are given and we therefore use following approximation:

$$\hat{R} \approx \max_{k: t'[k] < 2s} \{r'[k]\}. \quad (5.2)$$

Remark. Now we explain the choice of the constants. We limit the peak rate to be located within the first 2 s: If there is a token bucket traffic shaping, then from our experience (recall fig. 2.15) the level shift does not happen later than at 1 s + accounting for some reserve.

The thinning interval Δ_{\min} used in the open data is 100 ms. According to subsec. 2.3.3 we choose $nT = \Delta_{\min}$ such that the thinned sequence can be interpreted as noisy subsampling from the smoothed resampled non-thinned rate $T^s(n)[k]$. This justifies the approximation in eq. (5.2).

¹The index range $k \in \{n-L, \dots, 20n-L\}$ follows from the mapping in eq. (2.13).

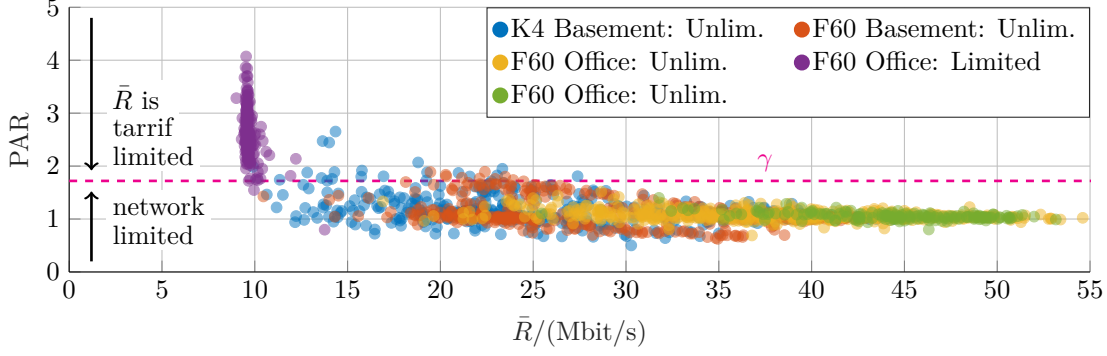


Figure 5.3: Scatter plot of peak to average ratio versus average rate. One mark in scatter plot corresponds to one test. Different colors distinguish different measurement scenarios. Traffic shaping is detected if $\text{PAR} \geq \gamma$.

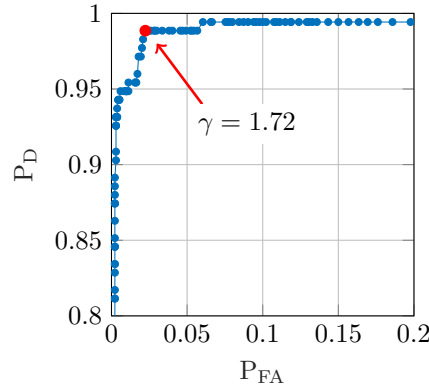


Figure 5.4: Empirical ROC (receiver operating characteristics). We picked $\gamma = 1.72$ (red point) since it is closest to perfect detection $(P_{\text{FA}}, P_{\text{D}}) = (0, 1)$. P_{D} denotes true positive rate, P_{FA} denotes false positive rate. Since we base ROC on empirical data we can't talk about detection probability and false alarm probability.

5.2.1 Controlled Measurements

First we apply the PAR to our controlled measurements. We took 300–400 static Open-RMBT TU measurements for every scenario. We tested two different device models (LG K4 and LG F60) at two different locations (office in the first floor and basement). In the basement we expect worse signal reception, therefore lower rate. In one measurement scenario we used tariff limited SIM card, in other scenarios tariff unlimited SIMs.

Results are shown in fig. 5.3. For every test we display the PAR versus average rate as one point in scatter plot. Such representation is particularly useful for representing sets of tests, because we can immediately read out the shaping rate on the x -axis at locations where we observe vertical peaks. We can immediately recognize, that in the “purple set” of tests the data rate was tariff limited with shaping rate of ≈ 9.6 Mbit/s.

As a reasonable threshold for traffic shaping detection we pick $\gamma = 1.72$ (based on receiver operating characteristics in fig. 5.4 constructed from our controlled measurements for which we know the ground truth) and detect tariff limitation if $\text{PAR} \geq \gamma$.

5.2.2 Open Data

In the case of RTR open data only C thinned sequences $(w'_c[k], t'_c[k])_{k \in \{1, \dots, K_c\}}$ with $C = 3$ (or $C = 1$ if connection is too slow, as explained in sec. 3.2) are available. Therefore we

ISP	Tests	Downlink			Uplink		
		TS tests	\bar{R} of all	\bar{R} without TS	TS tests	\bar{R} of all	\bar{R} without TS
A	1879	15 %	57.91	63.42	20 %	24.90	27.98
B	1128	6 %	80.55	83.95	7 %	35.47	35.92
C	943	2 %	53.24	53.72	6 %	30.94	31.29

Table 5.1: Number of tests, percentage of detected traffic shaped (TS) tests, average rate of all tests and average rate of non traffic shaped tests for ISP A, B, C in both, DL and UL.

first calculate the merged sequence $(w'[k], t'[k])_{k \in \{1, \dots, K\}}$ and then use approximation in eq. (5.2). The average rate is already given in the database so we don't have to calculate it. We applied this approach to LTE tests of three major Austrian mobile operators.

RTR open data provide network MCC/MNC and SIM MCC/MNC (part of IMSI) as well as network type. For every operator we pulled only tests conducted in LTE network with MCC = 232 (Austria) and MNC corresponding to given operator,² such that network MCC/MNC equals the SIM MCC/MNC in order to not consider roaming. We took tests in the time range from 2016-08-01 00:00:01 to 2016-11-01 00:00:00. For ISP A we obtained 1879 results, for ISP B 1128 and for ISP C 943.

Scatter plots are shown in fig. 5.5. Numerical results are provided in table 5.1. The strongest evidence of traffic shaping was found in tests of ISP A in both directions, DL and UL. In DL we see multiple vertical lines which correspond to different shaping rates (probably different user tariffs). In case of ISP B there is some traffic shaping happening in DL, the evidence in UL is not very convincing. Finally for operator C the algorithm did not detect almost any traffic shaping, which however doesn't mean that ISP C is not limiting rate of its users.

The RTR's statistics [70] show only average rate of all tests fulfilling filter criteria (time, location, ISP, etc.). With additional traffic shaping detection we can compare only non traffic shaped tests, obtaining different average rates (tab. 5.1).

Limits of the PAR Method

The detection whether a tariff limitation is in place or not depends on a detectable level change at the beginning of the time series. If the operators use a more smooth traffic shaping method a reliable detection currently seems not possible.

Another problem is following: Let's imagine we find in the open data two tests with high signal strength (e.g. RSRP = -80 dBm) but poor average rate (e.g. 10 Mbit/s in DL). In one case the poor rate could be caused by tariff limitation in the other by user's crosstraffic (e.g. user watching YouTube video when conducting the test).

ISPs' Benchmark Detection and Optimization

More detailed analysis of traffic shaping in case of ISP C revealed, that data rates are higher (corresponding to rates of e.g. ISP A in tariff unlimited case) when running RTR-NetTest towards official RTR server than when running OpenRMBT TU towards our own server. We didn't observe such difference for IPS A and B.

This suggests that ISP C recognizes IP address of RTR's server and provides users with full rate in order to reach better results public statistics of RTR [70]. This highlights one of the challenges in crowdsourcing, as the clients and even the network can not be

²Some operators have multiple MNCs. We select the one which has the most database entries, the other MNCs show usually only few tests.

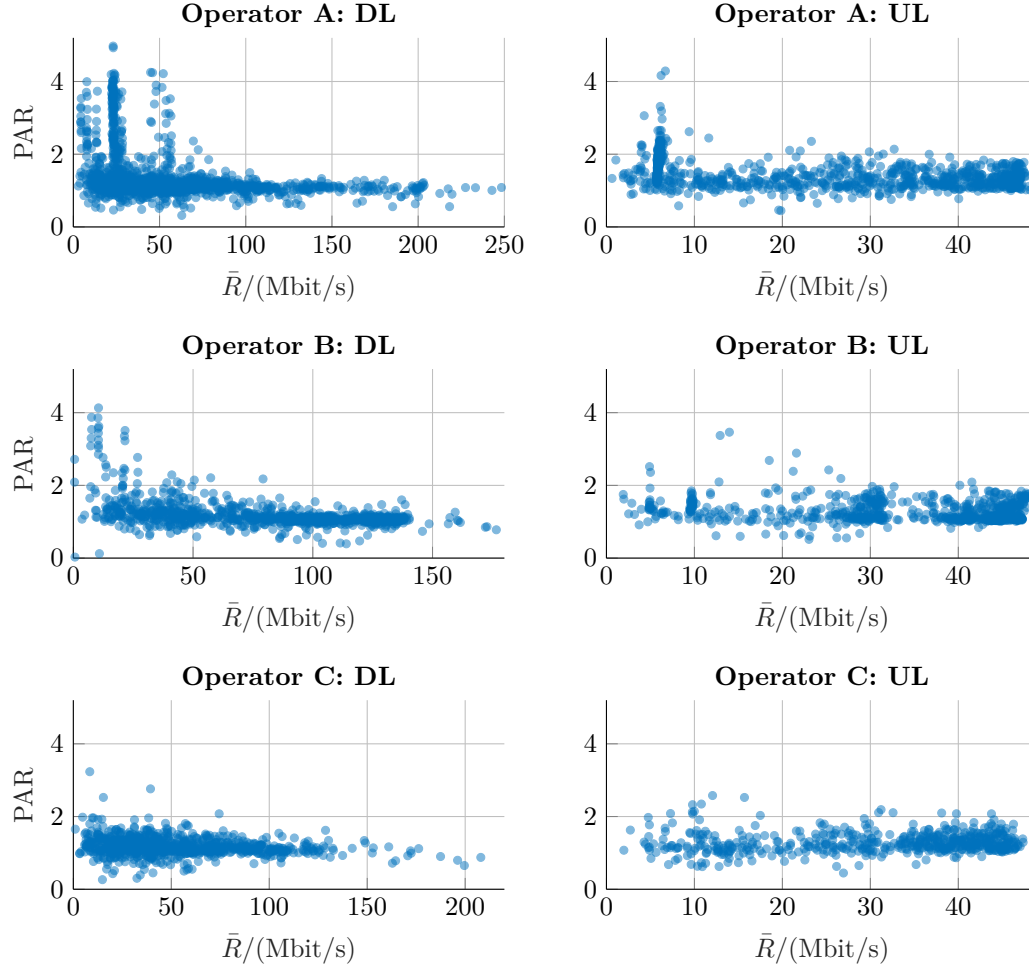


Figure 5.5: Comparison of PAR vs mean rate in DL and UL in LTE networks of three major Austrian mobile ISP.

trusted to behave neutral. Randomized test servers could be used to detect such traffic shaping.

5.3 Operator Benchmarking Using the RTR Data Set

In the previous section we considered only the shape of the data rate curve. From theory we expect that users with higher RSRP, which is also reported in open data,³ achieve higher throughput. Figure 5.6 shows mean rate \bar{R} versus RSRP as a 2D histogram of all LTE tests of ISP A. The histogram is logarithmic to achieve better visibility large range of values – color of each bin represents natural logarithm of number of tests with mean rate and mean RSRP falling into that bin. Bin size was chosen $2 \text{ dB} \times 2 \text{ (Mbit/s)}$. This is set empirical, based on the resolution seen in the data-set.

In order to analyze the maximum network performance offered by the mobile networks, we only consider the best tests (highest average rate) for every RSRP level and construct a “capacity curve” for each operator. To exclude spurious outliers we applied median filter and then found bin with highest \bar{R} and value > 0 (i.e. more than one test in bin due to logarithm) for every RSRP. Fig. 5.6 shows capacity curves (red) for different sizes

³Not for all tests, from unknown reason, maybe due to implementation. We only consider entries with valid signal strength information.

of median filter at histogram of ISP A. We picked the filter size 30×30 to get rid of the strong vertical lines between -90 and -60 dBm and above 160 Mbit/s. More comments on this will follow in subsec. 5.3.1.

We obtained capacity curves also for other two ISP in DL and UL. Result is shown in fig. 5.7. What is also plotted are mean data rates in DL for six different attenuation levels from previous measurements in reference cell of ISP A (subsec. 4.6.1). The results show that the reference measurement matched the data collected from the crowdsourced data sample for ISP A. Furthermore it shows significant differences between the operators. At the current status we cannot name the source of the difference. However, further analysis will target to analyze urban / rural areas, network deployment and interference analysis, e.g. note the different performance in uplink.

The reason why we select only the best tests for given signal strength is that we don't know all the details of single tests like presence of crosstraffic, tariff limitation in absence of rate overshoot, BS handover, speed of the UE, ... which can severely decrease the measured rate. This method has however disadvantage that it can be manipulated; see next subsection.

5.3.1 Filtering Spurious Test Entries

The high peaks observed in fig. 5.6 seem to originate from a process that is purely deterministic, e.g. caused by measurements in a static reference scenario. In case of standard measurement samples we would expect the variance to decrease for an increasing sample set. Here the opposite is the case. In fact the peaks' grow is an increasing sample set biasing the whole population.

The indications for the source are many tests performed at different constant RSRP levels – similar to our reference cell measurements.⁴ From our experience we hardly observe RSRP better than -80 dBm, even for rooftop measurements where we should have line of sight connection with base station. Another cause could be some buggy UE model which reports constant nonsense RSRP values.⁵

5.3.2 Conclusion

The presented lightweight metrics allow fast processing on large sets of tests. The analysis showed that it is crucial to choose proper filter criteria, like network MCC/MNC matching SIM MCC/MNC. On the other hand the publicly available benchmark data sets shouldn't be too simple – e.g. average rate of all LTE tests of ISP X together [70] – otherwise operator benchmarking can become just competition which operator is able to inject more tests with higher rate.

Based on comparison in fig. 5.7 we conclude that crowdsourced data can be used to obtain similar results as the measurements in reference network.

⁴We avoid influencing open data by using our own RMBT measurement and database server.

⁵What is also very suspicious is the bright “hot spot” at $\approx (-76$ dBm, 140 Mbit/s) in fig. 5.6.

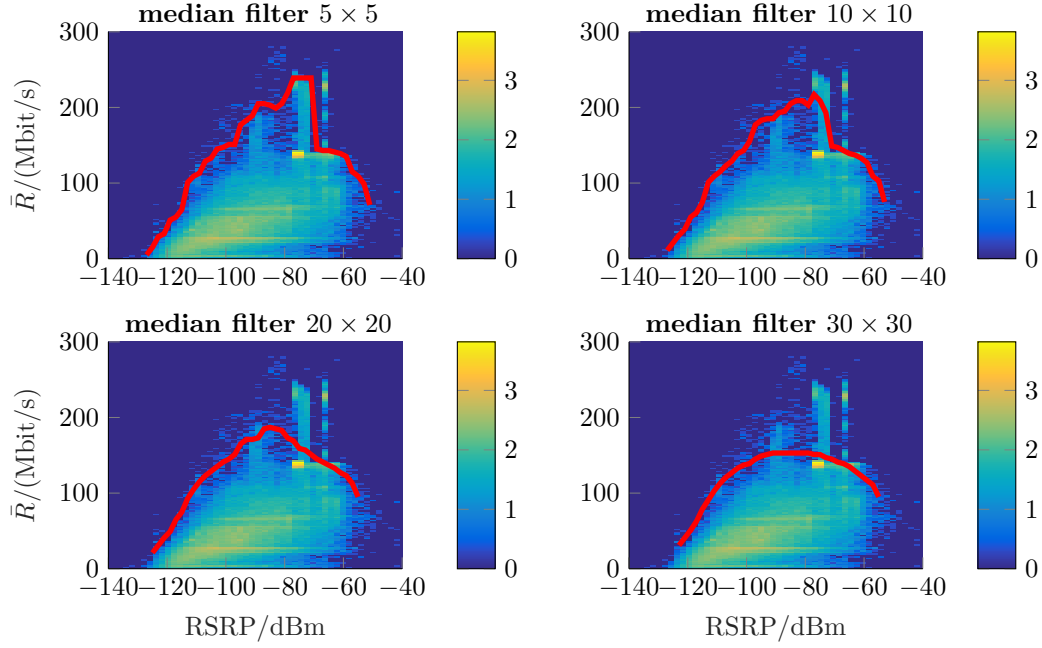


Figure 5.6: 2D histogram showing distribution of LTE tests of ISP A as a function of RSRP and mean rate \bar{R} . The red curve characterizes the highest average rate for given RSRP after using median filter to get rid of outliers.

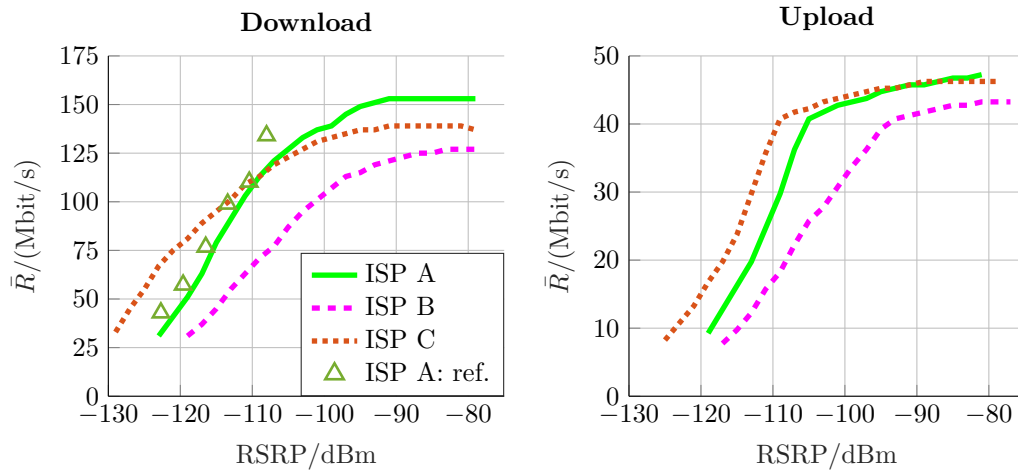


Figure 5.7: Comparison of “capacity curves” of different operators. The triangles in the left plot show measurements performed in unloaded reference cell of operator A.

Summary and Outlook

Summary

The concept of crowdsourced measurements allows service providers to outsource the expensive and time consuming task of performance measurements to the end user device. This approach is obviously appealing for the industry, though giving up the control on the experiment in favor. In this thesis we took a first step into the analysis of crowdsourced data sets in order to build up a view of the current status of the network. We try to answer the question whether it is possible to use crowdsourced data to meaningfully characterize properties of the network.

In the first chapter we presented a review of existing methods, discussed properties of TCP as well as the standards for benchmarking mobile networks. We identify that there is a trade-off between precision, service protocol and duration of the tests. We perform first measurements and verify that a single TCP connection is enough to fill the bandwidth delay product of the network under test. We also see that different applications (iPerf3 vs HTTP and FTP) deliver different results. Loosening the standards' conditions makes measurements more flexible but at the same time reduces reproducibility.

As we identify the need for repeatable measurements we have developed a measurement framework. The framework consists of an application part running on an Android smart phone, a server hosting the results and benchmark servers. The framework allows to collect measurements in a hybrid fashion, combining the idea of crowdsourced end-terminals with the planning of regular measurements intervals.

Performance measurements in networks are in general based on time-series of transmitted packets. In second chapter we established notation and proposed systematic methods for merging sequences of multiple connections of single user as well as resampling to equidistant time grid in order to compare time series of different users. We also discussed compression algorithm which is implemented in RTR's NetTest and makes the whole problem more challenging.

In the fourth chapter we analyzed controlled measurements, replicating collection of crowdsourced data in order to get more insight how the RTR's open data are produced. Loosening standards' requirements is partially compensated by possibility of large number of repeated measurements. With slight modifications of RTR NetTest we are able to obtain detailed results showing there is large potential in crowdsourced data.

In the fifth chapter we extend our analysis to the open data set and apply more lightweight metrics to gain overview of the network. The answer to the question is that open data have great potential but we need to be very careful which samples we select. We have to face many obstacles – RTR's compression algorithm dropping most of the cumulative volume samples; RTR-NetTest's unknown offset of individual connections causing data rate series oscillations which can be removed only with high computational expenses; ISPs performing automatized tests and ISPs switching off tariff limitation during connection to RTR's server and thus biasing the benchmarking process.

Outlook

There is room for further work left. This thesis provides only the ground. As a first step we can understand that the thinning process should be improved to allow us to reconstruct the original time series. We shall propose a modification to RTR.

Second the discrimination of data rate tests due to user tariff and due to network condition allow further analysis of the health and performance of a network under test.

Third the preprocessing of the data set enables further research on extrapolation, e.g. with Gaussian process regression in time, as well as root cause analysis of findings, e.g. via machine learning with deep neural networks.

Appendix A

Volumes and Rates

A.1 Alternative Merging Algorithm

We could think of a different merging strategy which does not use any interpolation. The volume samples of merged sequence are obtained by union of volume samples of individual connections (in case some volume samples have the same time location they are just summed up). It sounds more natural than algorithm in subsection 2.1.3. We will use subscript $_{\text{am}}$ to distinguish the alternative merging algorithm from the one presented in subsec. 2.1.3.

The alternative merging algorithm is used for example in [38]: cumulative volume sample of merged sequence is given by “the sum of all bytes transferred since the start of the test phase.” The sum of all bytes is here meant on all connections together.

The merged cumulative volume sequence is initialized as follows:

$$\begin{aligned} t_{\text{am}}[0] &:= 0, \\ w_{\text{am}}[0] &:= 0, \\ K_{\text{am}} &:= 0. \end{aligned}$$

In every iteration we find the smallest unused time location t^* among all time locations of all connections:

$$\begin{aligned} k_{\min,c} &:= \arg \min_{k \in \{1, \dots, K_c\}} t_c[k] \quad \text{s.t.} \quad t_c[k] \text{ unused}, \quad \forall c \in \{1, \dots, C\}; \\ c^* &:= \arg \min_{c \in \{1, \dots, C\}} t_c[k_{\min,c}]; \\ t^* &:= t_{c^*}[k_{\min,c^*}]; \end{aligned}$$

where the word “unused” means that $t_c[k]$ has not been assigned to t^* in any previous iteration. The volume received at the time t^* on the c^* -th connection is denoted by v^* :

$$v^* := w_{c^*}[k_{\min,c^*}] - w_{c^*}[k_{\min,c^*} - 1].$$

Now there are two possibilities:

1. Either $t_{\text{am}}[K_{\text{am}}] = t^*$: In this case the time sample t^* already exists in the merged sequence and we just need to increase the corresponding cumulative volume sample of the merged sequence by the volume received at time t^* on the c^* -th connection:

$$w_{\text{am}}[K_{\text{am}}] += v^*.$$

2. Or $t_{\text{am}}[K_{\text{am}}] \neq t^*$: In this case the time location t^* is not contained in the merged sequence and we have to add it:

$$\begin{aligned} K_{\text{am}} &+= 1; \\ t_{\text{am}}[K_{\text{am}}] &:= t^*; \\ w_{\text{am}}[K_{\text{am}}] &:= w_{\text{am}}[K_{\text{am}} - 1] + v^*. \end{aligned}$$

We continue with next iteration until there are no unused samples left.

The algorithm above is described in terms of cumulative volumes. It can be easily described also in terms of volume samples of separate connections which are placed to a common time axis (graphical representation in fig. A.1, top right).¹

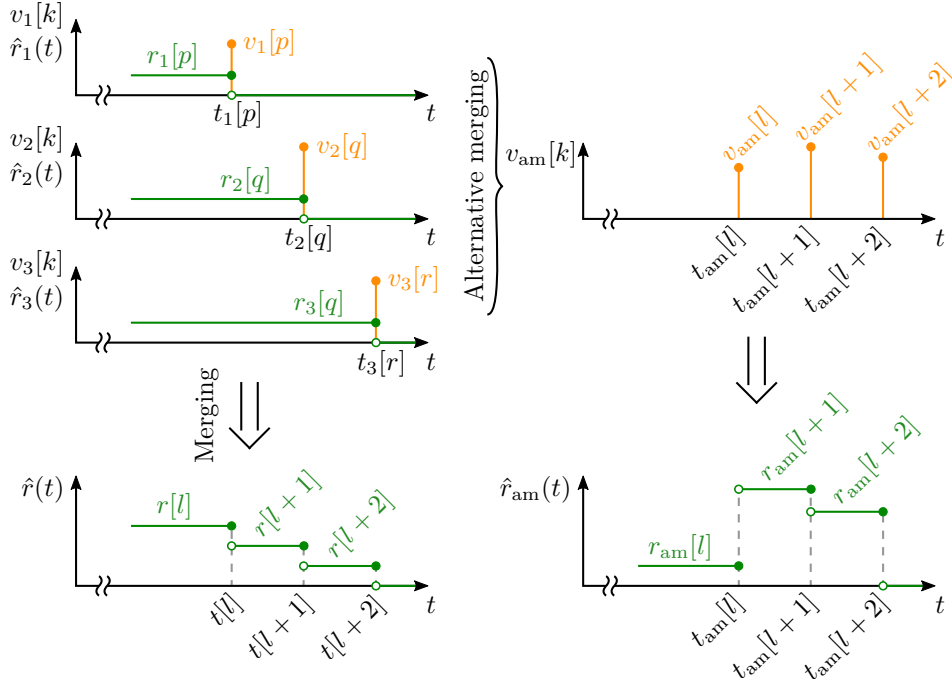


Figure A.1: Comparison of two different merging algorithms. The first merging algorithm is based on sum of time-continuous rates of individual connections – equations (2.5), which leads to the expression (2.7). The alternative merging algorithm is based on an idea that we stack volume samples of all connections to a single sequence without modifying their time locations.

A.2 Comparison of Both Merging Algorithms

The time locations are identical for both algorithms: $t_{\text{am}}[k] = t[k]$, $\forall k$. The fig. A.1 proves that cumulative volume samples (and therefore also rate samples) are in general different, $w[k] \neq w_{\text{am}}[k]$. There are several reasons why we preferred the first merging algorithm:

- **Linearity:** The sum of time-continuous rates $\hat{r}_c(t)$ of individual connections is equal to the total rate $\hat{r}(t)$. This is not true for the alternative merging algorithm.
- **TCP connection example:** For every connection the k -th data rate sample is calculated as $v_c[k]/(t_c[k] - t_c[k - 1])$, i.e. the data rate is assumed to be constant in the

¹If there would be two samples with the same time, e.g. $t_{c_1}[k_1] = t_{c_2}[k_2]$, we would have to sum both volume samples $v_{c_1}[k_1] + v_{c_2}[k_2]$ before placing them to the merged sequence.

interval $(t_c[k-1], t_c[k])$ in which we don't observe any data volume samples. In ideal case, the spaces $t_c[k] - t_c[k-1]$ between consecutive samples $v_c[k]$ would correspond to RTT. By merging we increase number of time locations and thus create new, shorter intervals. If we apply alternative merging algorithm, then we divide data volume by these shorter intervals when calculating data rate. The shorter intervals, which lead to higher data rate peaks, don't represent any RTT as in case of separate connections.

- Thinning: If we don't know all samples (like in case of thinning in sec. 2.3) it is reasonable to assume that the rate was constant in a given interval – e.g. we know $w_c[k]$ at time $t_c[k]$ and $w_c[k+m]$ at time $t_c[k+m]$ but nothing in between; we assume that rate is equal to $(w_c[k+m] - w_c[k]) / (t_c[k+m] - t_c[k])$ within whole interval. We can add this rate to the rates of other connections.

If we use alternative merging instead, it can happen that there is a connection d with a sample $w_d[l]$ at time $t_d[l]$ which is just a little bit smaller than $t_c[k+m]$, resulting in very short interval and very high rate – in other words: instead of distributing the difference $w_c[k+m] - w_c[k]$ to the whole interval $(t_c[k], t_c[k+m])$ uniformly, we shrink it to possibly much smaller interval $(t_d[l], t_c[k+m])$.

⇒ In alternative merging algorithm the sample positions of one connection impact the rate of other connections!

A.2.1 Data Rate Examples

In this subsection we compare both algorithms using real data. We took results of Opne-RMBT TU with $c = 3$ TCP connections and test duration of $\Delta t = 7$ s.

Qualitatively we can say that for both algorithms the most of the samples are lower or equal to 200 Mbit/s (fig. A.3). Also the shape of irregularly wide bins is similar, as can be seen in the detail in the right part of the figure. In our example $\hat{r}_{\text{am}}(t)$ has more peaks which are much higher compared to $\hat{r}(t)$. If we zoom out (fig. A.2, y -axis is in Gbit/s) we see that $\hat{r}_{\text{am}}(t)$ has peaks in order of tens of Gbit/s, whereas peaks of $\hat{r}(t)$ in fig. A.3 are not larger than 400 Mbit/s.

To get some feeling how different or similar both results are, we compare the MSD between them. Plugging the merged data rates $r_{\text{am}}[k]$ and $r[k]$ into eq. (A.2) we obtain $\text{MSD} \approx 3287.83 \text{ (Mbit/s)}^2$. As said the rate $r[k]$ reaches maximum rates of ≈ 400 Mbit/s whereas $r_{\text{am}}[k]$ shows peaks up to ≈ 90 Gbit/s. These peaks are very narrow and therefore they are weighted with smaller factor $(t[k] - t[k-1])$, the MSD is however still quite large.

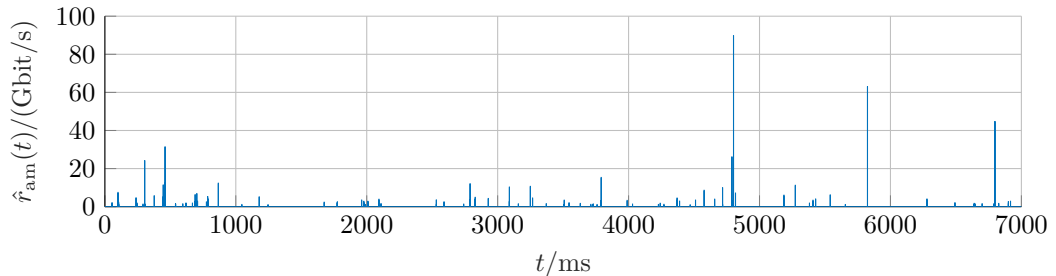


Figure A.2: Data rate $\hat{r}_{\text{am}}(t)$ produced by alternative merging algorithm. The y -axis is represented in Gbit/s with just few very narrow and very high peaks. In the intervals where the data rate appears to be zero we can still observe some nonzero values after zooming in (fig. A.3).

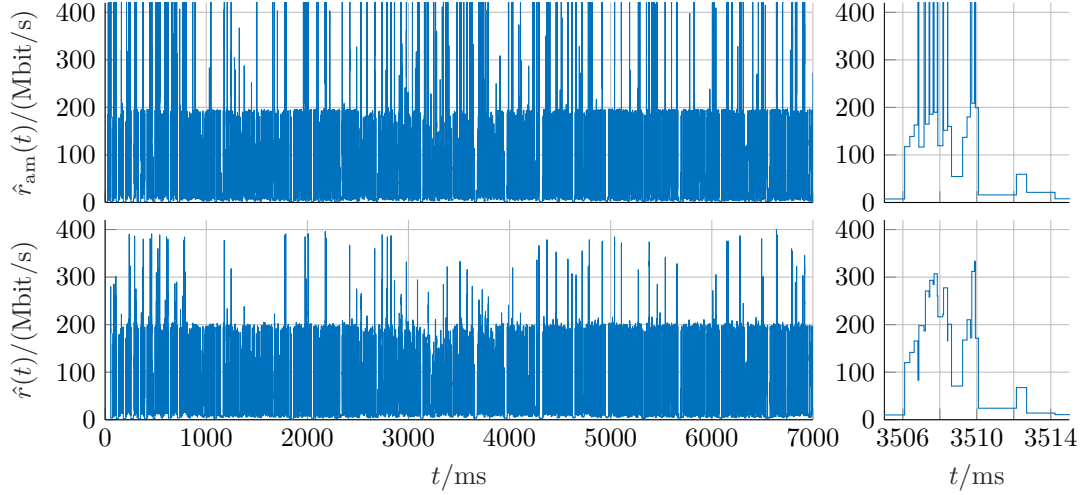


Figure A.3: Comparison of data rate $\hat{r}_{am}(t)$ (top) and $\hat{r}(t)$ (bottom).

Resampling and Smoothing, Thinning

After resampling with $T = 1$ ms we obtain $\tilde{r}, \tilde{r}_{am}$ with $\Delta_{\max} \approx 60.26$ Mbit/s (see def. A.3.2 and eq. (A.5)) and $\text{MSD} \approx 57.68 (\text{Mbit/s})^2$. Comparison of ${}_T s^{(n)}$ and ${}_T s_{am}^{(n)}$ – average data rates in intervals of $n = 101$ ms considering 101 different offsets mT – results in $\Delta_{\max} \approx 2.97$ Mbit/s and $\text{MSD} \approx 0.05 (\text{Mbit/s})^2$. We thus understand that with averaging intervals large enough the impact of peaks can be suppressed due to their narrowness and sparsity and we obtain similar results for both algorithms. This is no proof, but it again suggests that the smoothed rate ${}_T s^{(n)}$ is a suitable quantity for representing the data rate.

In fig. A.4 in the upper plot we visualize ${}_T s_{am}^{(n)}[k]$ obtained by smoothing ($n = 101$) of resampled ($T = 1$ ms) total rate obtained by alternative merging $r_{am}[k]$; and time-continuous rate $\hat{r}'_{am}(t)$ obtained by thinning of cumulative volume sequences, merging them using alternative merging and calculating rate out of them.

In the lower plot we visualize same quantities, with the only difference that the regular merging algorithm was used. We see that alternative merging leads to very high peaks when applied to thinned sequences, because it artificially creates very short intervals as explained in sec. A.2.

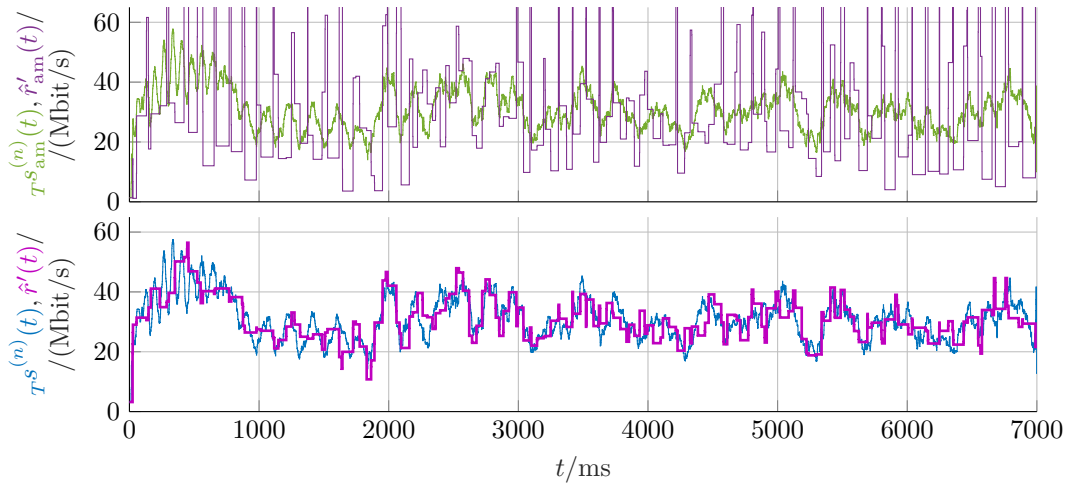


Figure A.4: Smoothed rates after merging and resampling; and thinned rates after thinning and merging. We plot discrete samples such that the k -th one corresponds to $t = kT$.

A.3 Mean Squared Difference, Maximum Difference

Definition A.3.1. Given two functions $f, g: \mathbb{R} \rightarrow \mathbb{R}$ with support $[0, \Delta t]$, we define mean squared difference (MSD) as follows:

$$\text{MSD}(f, g) \triangleq \frac{1}{\Delta t} \int_0^{\Delta t} (f(t) - g(t))^2 dt. \quad (\text{A.1})$$

Because time locations are identical for both merging algorithms, $t_{\text{am}}[k] = t[k]$,

$$\begin{aligned} \text{MSD}(\hat{r}_{\text{am}}, \hat{r}) &= \frac{1}{\Delta t} \int_0^{\Delta t} (\hat{r}_{\text{am}}(t) - \hat{r}(t))^2 dt = \\ &= \frac{1}{\Delta t} \sum_{k=1}^K (t[k] - t[k-1]) (r_{\text{am}}[k] - r[k])^2, \end{aligned} \quad (\text{A.2})$$

where $t[K] = \Delta t$ and $t[0] = 0$.

Assuming $\Delta t = nT_s$, $n \in \mathbb{N}^+$, and any two quantities $x_1[k]$, $x_2[k]$ (and their time-continuous representations $\hat{x}_1(t)$, $\hat{x}_2(t)$) obtained by resampling with period T we can further simplify the expression A.2:

$$\text{MSD}(x_1, x_2) = \frac{1}{nT} \int_0^{nT} (\hat{x}_1(t) - \hat{x}_2(t))^2 dt = \frac{1}{n} \sum_{k=1}^K (x_1[k] - x_2[k])^2. \quad (\text{A.3})$$

Definition A.3.2. Given two function $f, g: \mathbb{R} \rightarrow \mathbb{R}$ we define their maximum difference

$$\Delta_{\max}(f, g) \triangleq \max_t |f(t) - g(t)|. \quad (\text{A.4})$$

Assuming time-continuous representations $\hat{x}_1(t)$, $\hat{x}_2(t)$ based on samples $(x_1[k], t_1[k])$, $(x_2[k], t_2[k])$ which have identical time locations, i.e. $t_1[k] = t_2[k] \forall k$, the maximum difference can be rewritten in terms of time discrete samples:

$$\Delta_{\max}(\hat{x}_1, \hat{x}_2) = \max_t |\hat{x}_1(t) - \hat{x}_2(t)| = \max_k |x_1[k] - x_2[k]|. \quad (\text{A.5})$$

List of Figures

1.1	State diagram of TCP congestion control algorithm. Source: [14], figure 3.52.	8
1.2	Wireshark trace of server's RWND during an iPerf3 UL test.	14
1.3	Wireshark trace of UE's RWND during an iPerf3 DL test.	15
1.4	Wireshark trace of TCP RTT during an iPerf3 UL test.	16
1.5	Wireshark trace of TCP throughput (right vertical axis) of the same test as in fig. 1.4. (Moving average with window size of 1 s.)	16
1.6	Comparison of TCP throughputs of iPerf3 UL test with a single TCP connection (red solid) and with five parallel connections (blue dashed). Each subplot is zoomed to different test phase.	17
1.7	The same 1- and 5-connection tests as in fig. 1.6, this time smoothed using moving average with windows size of 1 s in order to visualize the throughput trend. Left: whole test; duration 30 s. Right: detail showing the initial ramp-up phase, including reference (see fig. 1.8).	17
1.8	We fit linear and quadratic functions (right plot) to the smoothed throughput (fig. 1.7, right) in order to represent the ramp-up phase in terms of continuous functions. The left plot shows, how these fitted functions look like without smoothing.	17
1.9	RTT of a single TCP connection during 5-connection iPerf3 UL test.	18
1.10	CWND size during the 1-connection iPerf3 UL test. Green line = RWND size.	19
1.11	CWND size of a single TCP connection during the 5-connection iPerf3 UL test. Green line = RWND size.	19
1.12	Empirical CDF plots of iPerf3, HTTP and FTP throughput in DL and UL. For iPerf3 we took all configurations together here.	20
1.13	ECDF of iPerf3 throughput using different number of TCP connections.	20
2.1	$\hat{r}(t)$ is the time-continuous estimate of the true rate $r(t)$. The k -th time discrete sample $r[k]$ represents the average of the true rate in time interval $\mathcal{I}[k] = (t[k-1], t[k]]$.	22
2.2	"Correct" resampling corresponds to rebinning of the time-continuous estimate $\hat{r}(t)$ in such a way that the mean of resampled rate $\tilde{r}(t)$ is equal to the mean of $\hat{r}(t)$ in every time interval $[(k-1)T, kT]$.	23
2.3	An example of "wrong," direct resampling $\check{r}[k] = \hat{r}(kT)$. In this example the mean rate in the interval $[0, T]$ is significantly larger after resampling.	24
2.4	Merging rates of two connections: $\hat{r}(t) = \hat{r}_1(t) + \hat{r}_2(t)$. First connection can be represented by K_1 pairs $(t_1[k], r_1[k])$, second connection by K_2 pairs $(t_2[k], r_2[k])$ and the merged sequence by K pairs $(t[k], r[k])$.	26
2.5	Two equivalent algorithms for calculating the total resampled cumulative volume sequence. The merging is defined by eq. (2.5) and can be expressed as in eq. (2.7).	28

2.6	Representation of algorithms from fig. 2.5 in terms of stepwise continuous rates. Resampling of rate $\hat{r}(t) \rightarrow \tilde{r}(t)$ is given by def. 2.1.8 and 2.1.9. For time-continuous rates the merging operation is replaced by summation, which is convenient for mathematical description but not suitable for discrete implementation.	28
2.7	Illustration showing the worst case error. Left: binning of the true rate $r(t)$ with offset τ_1 . Right: binning with offset τ_2 . Bin size T is constant but otherwise arbitrary. We obtain the worst case error for $ \tau_1 - \tau_2 = T/2$. The yellow area corresponds to $T \cdot \frac{\tau_1, \tau_2}{T} \varepsilon[i + 1]$. Thus $\frac{\tau_1, \tau_2}{T} \varepsilon[i + 1] = \frac{r_{\max} - r_{\min}}{2}$	30
2.8	The upper plot shows $\frac{\tau}{T} s^{(n)}[k]$ (gray) – the smoothed version of $\frac{\tau}{T} r[k]$. By subsampling of $\frac{\tau}{T} s^{(n)}[k]$ according to eq. (2.13) we obtain n different binnings $\frac{\tau'}{nT} r[k]$ with $\tau' \in \{\tau, \tau + T, \dots, \tau + (n - 1)T\}$. Here we illustrate two different subsamplings: with $\tau' = \tau + m_1 T$ (blue) and $\tau' = \tau + m_2 T$ (red), where $m_1 = 0$, $m_2 = (n - 1)/2$ and $n = 101$. The upper plot shows discrete representations $\frac{\tau'}{nT} r[k]$ and the lower plot corresponding time-continuous representations $\frac{\tau'}{nT} r(t)$. The samples in the upper plot are aligned with the lower plot such that k -th sample of $\frac{\tau}{T} s^{(n)}[k]$ corresponds to time $t = kT$ – the time locations of samples in upper plot then correspond to the centers of the time-continuous bins in the lower plot.	32
2.9	Deriving the worst case uncertainty after convolution with rectangular window of size n . Even though the bin duration is now nT , we still observe the worst case uncertainty for the offset difference $ \tau_1 - \tau_2 = T/2$	33
2.10	Representation of thinned sequence $r'[k]$ as noisy subsampling from $\frac{\tau}{T} s^{(n)}[k]$ (upper plot) and $\hat{r}(t)$ – time-continuous representation of $r'[k]$ (lower plot).	36
2.11	The non-zero part of the signal $\frac{\tau}{T} s_c^{(n)}[k] = (a + b \cdot \sin(\phi(kT))) \cdot \text{rect}\left[k - \frac{K}{2}; \frac{K}{2}\right]$ (left) and its spectrogram (right). $T = 1$ ms, $t = kT$	37
2.12	Objective function of minimization problem in eq. (2.25) using model from subsec. 2.4.1. Note: var in the axes labels is shorthand for variance of $\frac{\kappa}{T} s^{(n)}$ calculated for $k \in \{0, \dots, K\}$. With our resampling period $T = 1$ ms, the shift of κ samples corresponds to time shift of κ ms.	39
2.13	Minimum variance solution $\frac{\hat{\kappa}}{T} s^{(n)}[k]$ obtained from minimization problem in eq. (2.25) by exhaustive search $(\kappa_2, \kappa_3) \in \{-500, \dots, 500\}^2$	39
2.14	Variance of $\frac{\tau}{T} s^{(n)}[k]$ when considering phase shifts ϕ_2, ϕ_3 instead of index shifts κ_2, κ_3 . The phase shift ϕ_1 was fixed to 0. The obtained pattern is 2π -periodic and we would see similar results also for different phase functions $\phi(t)$	40
2.15	An example of traffic shaping. $\hat{C} \approx 30.38$ Mbit/s is the estimate of peak rate, $\hat{\rho} \approx 9.65$ Mbit/s is the estimate of token generation rate and $\hat{\sigma} \approx 14.38$ Mbit ≈ 1.8 MB (the yellow area) is the estimate of the token bucket size.	41
3.1	The CMPT framework: Configuration is specified via web interface. The UEs running CMPT application obtain configurations from the CMPT server, perform various measurements targeting different servers and upload measurement results back to the CMPT server. Author of this picture: Philipp Svoboda.	42
3.2	CMPT application menu (left) and the main screen (right).	43
3.3	CMPT application screens. Left: List of available WLANs (experimental feature). Middle: Overview of test configurations. Right: Miscellaneous.	44
3.4	CMPT scheduling (upper block) and passive monitoring (lower block).	46

3.5	Screenshot of the CMPT settings web interface.	47
3.6	An example of different test periods. ICMP ping period is 5 min, FLARP period is 10 min and iPerf3 period is 20 min. This means that ICMP ping will be executed in every block, FLARP in every second and iPerf3 in every fourth. Open-RMBT test period was set to 0, which means disabled.	48
3.7	Example of two UEs with identical settings of test periods. The only difference is that the block offset of the UE1 was set to 0 min and the block offset of the UE2 was set to 5 min.	48
3.8	A screenshot showing part of JSON document uploaded by CMPT application to the CouchDB database.	48
3.9	Wireshark's packet capture on the server side showing the phases 2–6 of the RMBT's test. The time on the x -axis is relative to the start of the packet capture.	53
3.10	Average data rate of c -th connection in an interval $[0, \Delta t]$ is calculated as $\bar{R}_c = \int_0^{\Delta t} \hat{r}_c(t) dt$. It is equal to $(w_c[K_c - 1] + \text{c.t.})/\Delta t$, where c.t. stands for a correction term corresponding to the area marked with gray color.	53
4.1	Comparison of rate reported by Wireshark (green) with rate reported by RMBT – before shift (red) and after shift of individual connections (purple). Top: Rate R with $T = 1$ ms. Middle: Smoothed with rectangular window of size 101 ($g_{101}[k]$). Bottom: Smoothed with quadratic window ($*g_{101}[k]$ three times).	58
4.2	Detail of first 200 ms of RMBT's data rate test. Data rates are plotted for every connection separately. Upper plot: The resampled rates R_k of individual RMBT's connections. Middle plot: Connections 1 and 3 are shifted in order to match the Wireshark's data rates. Upper and middle plot: Time zero corresponds to beginning of the DL test as reported by RMBT. Lower plot: Time zero corresponds to the beginning of the packet capture.	58
4.3	Measurement setup 2. LTE UE tethers an Internet connection to the laptop on which we run RMBT test and capture packets on the USB interface. The laptop also captures packets, which are mirrored from the server by a switch, on the Ethernet interface. Solid lines show the path the packets have to travel during the data rate test. Connection drawn as a dashed line is used just to capture packets also on the server side.	59
4.4	Wireshark's captures on the client (green) and server side (blue). Upper plot: R , $T_s = 1$ ms. Lower plot: S , quadratic window ($3 \cdot \text{span } 101$; used to clearly visualize trends despite zoomed-out x -axis). In upper representation we can clearly see time boundaries of DL pretest, latency test, DL test, etc. Smoothing introduces certain broadening, on the other hand it clearly shows the trend – we can thus recognize that mean data rate during DL was ≈ 30 Mbits. Times on x -axis are relative with respect to the capture beginning.	60
4.5	CDV vs time during RMBT DL test. We see that client receives more data than server sends. Samples reported by RMBT end around $t = 7$ s and already CDV trace indicates some oscillations. The different slopes of RMBT's and Wireshark's CDV are caused by header overhead.	60
4.6	Comparison of RMBT's DL test. Red: S (quadratic window) based on CDV samples reported by RMBT. Green: S based on Wireshark's capture. The RMBT trace is shifted in time in order to match the DL test captured by Wireshark. The Wireshark trace contains also DL pretest.	61

4.7	Comparison of RMBT's UL test. Red: S (quadratic window) based on CDV samples reported by RMBT. Blue: S based on Wireshark's capture. Here we compare RMBT with what the server is receiving, because RMBT's samples are collected on the server side during UL test and then transmitted back to client, see phase 6 in subsec. 3.2.1. RMBT trace is shifted in order to match the UL test captured by Wireshark. The Wireshark trace contains also UL pretest and results upload.	61
4.8	Smoothed rate on connection $c = 1$ in an RMBT DL test consisting of $C = 3$ connections.	62
4.9	Smoothed rate of an RMBT DL test consisting of only one connection. . . .	62
4.10	An example of mean squared difference between $R[k - \lambda]$ and $R_{\text{ref}}[k]$ as a function of shift λ , which shows that MSD is very sensitive to relative offset between the merged rate and the reference.	64
4.11	Left: Spectrogram of the merged rate $R[k]$. Right: Spectrogram of $\hat{\kappa}R[k]$ with $\hat{\kappa}$ minimizing the MSD between $\kappa R[k]$ and reference $R_{\text{ref}}[k]$	64
4.12	Mean squared difference w.r. to reference as a function of connection-shifts κ_2 and κ_3 . The global optimum is $\kappa_2 = -31$, $\kappa_3 = 1$, corresponding to -31 ms and 1 ms in time-continuous representation.	65
4.13	Variance of the resampled rate $R[k]$ as a function of shifts of 2nd and 3rd connection. The global optimum is $\kappa_2 = -36$, $\kappa_3 = -2$	65
4.14	Total power in band 2–15 Hz as a function of connections shifts. The global optimum is $\kappa_2 = -31$, $\kappa_3 = 2$	65
4.15	BS = before shift, AS = after shift. In first three rows we see resampled smoothed shifted rates $\kappa_c S_c$ and resampled thinned shifted rates $\kappa_c R_c$ of all three connections. The last row shows the total smoothed rate $\hat{\kappa}S$ after shift which removes oscillations. For the sum of thinned rates (magenta = after shift, orange = before shift) we don't see such improvement as in fig. 4.1 for the non-thinned rates.	67
4.16	Smoothed rates S_c (gray) of individual connections, the corresponding spectrograms of $R_c[k]$ and thinned rates R'_c (orange) represented as noisy subsampling from S_c . The red line in the top spectrogram shows which data points we used for the exponential fit to construct the model in sec. 2.4.1. . .	67
4.17	Example of smoothed rate S representing 101 different binnings with bin size of 101 ms in presence of token bucket traffic shaping. The different binnings are obtained by index mapping in theorem 5, subsection 2.2.3. . .	69
4.18	101 estimates of peak rate C based on 101 different binnings of rate R (left). Level shift was detected in 76 cases – for these different estimates of shaping rate ρ (middle) and bucket size σ (right) are obtained.	69
4.19	Histograms of estimates of 151 tests. For every bin size n we obtain n estimates per test, i.e. $151n$ estimates in total – this explains different heights of distributions.	69
4.20	Measurements in reference cell between May 10, 2017 and May 16, 2017. . .	71
4.21	For every RMBT measurement we calculate the smoothed rate $S[k]$. The figure shows 2D histogram (left) of all tests at attenuation level 1 (RSRP ≈ -122.7 dBm). The right subplots show histogram of all test samples. The two modes in DL indicate usage of two different modulation schemes.	71
4.22	Averaged curves at attenuation level 5 (RSRP ≈ -110.5 dBm).	72
4.23	One week of measurements (September 12 – September 19, 2016) in Kindberg, Austria in a live LTE network. Black lines represent trends from figures 4.24–4.26.	74
4.24	Weekly trend of RSRP for two nearest base stations.	75

4.25	Weekly trend of RSRQ for two nearest base stations.	75
4.26	Weekly trend of UL and DL data rates measured by iPerf3 and RMBT. . .	75
5.1	RTR-NetTest spatial distribution example. Source: internal software tool of Mobile Comm. group, Institute of Telecommunications, TU Wien.	78
5.2	Number of RTR-NetTest tests started (aggregate 2014–2015) as a function of time of day. Source: [69].	78
5.3	Scatter plot of peak to average ratio versus average rate. One mark in scatter plot corresponds to one test. Different colors distinguish different measurement scenarios. Traffic shaping is detected if $\text{PAR} \geq \gamma$	79
5.4	Empirical ROC (receiver operating characteristics). We picked $\gamma = 1.72$ (red point) since it is closest to perfect detection $(P_{\text{FA}}, P_{\text{D}}) = (0, 1)$. P_{D} denotes true positive rate, P_{FA} denotes false positive rate. Since we base ROC on empirical data we can't talk about detection probability and false alarm probability.	79
5.5	Comparison of PAR vs mean rate in DL and UL in LTE networks of three major Austrian mobile ISP.	81
5.6	2D histogram showing distribution of LTE tests of ISP A as a function of RSRP and mean rate \bar{R} . The red curve characterizes the highest average rate for given RSRP after using median filter to get rid of outliers.	83
5.7	Comparison of “capacity curves” of different operators. The triangles in the left plot show measurements performed in unloaded reference cell of operator A.	83
A.1	Comparison of two different merging algorithms. The first merging algorithm is based on sum of time-continuous rates of individual connections – equations (2.5), which leads to the expression (2.7). The alternative merging algorithm is based on an idea that we stack volume samples of all connections to a single sequence without modifying their time locations. . .	87
A.2	Data rate $\hat{r}_{\text{am}}(t)$ produced by alternative merging algorithm. The y -axis is represented in Gbit/s with just few very narrow and very high peaks. In the intervals where the data rate appears to be zero we can still observe some nonzero values after zooming in (fig. A.3).	88
A.3	Comparison of data rate $\hat{r}_{\text{am}}(t)$ (top) and $\hat{r}(t)$ (bottom).	89
A.4	Smoothed rates after merging and resampling; and thinned rates after thinning and merging. We plot discrete samples such that the k -th one corresponds to $t = kT$	89

List of Tables

3.1	List of CMPT tasks.	45
3.2	Arguments which can be used when executing Open-RMBT TU.2.2.12. . .	54
3.3	An overview of different Open-RMBT compilations comparing duration and number of parallel TCP connections of DL and UL data rate test.	55
4.1	Performance comparison of MSD, variance of non-smoothed rate $R[k]$ and power in the frequency band 2–15 Hz.	66
4.2	Means and standard deviations (don't confuse with token bucket size σ) of RSRP (in dBm) and RSRQ (in dB) for the six attenuation levels shown in fig. 4.20.	72
4.3	Means and standard deviations in Mbit/s for DL and UL mean rate \bar{R} of RMBT and iPerf3. In UL we give only values up to 4th attenuation level, at which the UE's limit is already reached.	72
5.1	Number of tests, percentage of detected traffic shaped (TS) tests, average rate of all tests and average rate of non traffic shaped tests for ISP A, B, C in both, DL and UL.	80

References

- [1] M. Hirth, T. Hofeld, and P. Tran-Gia, “Anatomy of a crowdsourcing platform – using the example of microworkers.com,” *Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, 2011.
- [2] F. P. Tso, J. Teng, W. Jia, and D. Xuan, “Mobility: A double-edged sword for HSPA networks,” *Proceedings of MobiHoc '10*, 2010.
- [3] M. Tomala, I. Keskitalo, G. Bodog, and C. Sartori, *LTE Self-Organising Networks (SON)*. John Wiley & Sons, Ltd, 2011.
- [4] “Universal mobile telecommunications system (UMTS); LTE; Universal terrestrial radio access (UTRA) and evolved universal terrestrial radio access (E-UTRA); radio measurement collection for minimization of drive tests (MDT); Overall description; Stage 2,” *3GPP TS 37.320 v11.1.0*, 2012.
- [5] “RTR-NetTest | Open Data.” <https://www.netztest.at/en/Opendata>. Accessed: 2017-03-09.
- [6] B. Constantine, G. Forget, R. Geib, and R. Schrage, “Framework for TCP throughput testing,” *IETF RFC 6349*, 2011.
- [7] M. Allman, V. Paxson, and E. Blanton, “TCP congestion control,” *IETF RFC 5681*, 2009.
- [8] J. Postel, “Transmission control protocol,” *IETF RFC 793*, 1981.
- [9] “LTE.” <http://www.3gpp.org/technologies/keywords-acronyms/98-lte>. Accessed: 2017-11-08.
- [10] “CellSignalStrengthLte | Android Developers.” <https://developer.android.com/reference/android/telephony/CellSignalStrengthLte.html>. Accessed: 2017-11-08.
- [11] “LTE; Evolved universal terrestrial radio access (E-UTRA); Physical layer; Measurements,” *3GPP TS 36.214 v12.3.0*, 2016.
- [12] “LTE; Evolved universal terrestrial radio access (E-UTRA); Physical layer procedures,” *3GPP TS 36.213 v8.8.0*, 2009.
- [13] “RSRQ to SINR relation.” <https://www.laroccasolutions.com/164-rsrq-to-sinr/>. Accessed: 2017-11-08.
- [14] J. F. Kurose and K. W. Ross, *Computer networking: a top-down approach*. Pearson Education, 6th ed., 2014.
- [15] H. Holma and A. Toskala, *LTE for UMTS: Evolution to LTE-Advanced*. John Wiley & Sons, 2nd ed., 2011.

- [16] D. Borman, B. Braden, V. Jacobson, and R. Scheffenegger, "TCP extensions for high performance," *IETF RFC 7323*, 2014.
- [17] M. Mathis and J. W. Heffner, "Packetization layer path MTU discovery," *IETF RFC 4821*, 2007.
- [18] M. Allman, S. Floyd, and C. Partridge, "Increasing TCP's initial window," *IETF RFC 3390*, 2002.
- [19] T. Henderson, S. Floyd, A. Gurtov, and Y. Nishida, "The NewReno modification to TCP's fast recovery algorithm," *IETF RFC 6582*, 2012.
- [20] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP selective acknowledgment options," *IETF RFC 2018*, 1996.
- [21] "BIC and CUBIC | Networking Research Lab." <https://research.csc.ncsu.edu/netsrv/?q=content/bic-and-cubic>. Accessed: 2017-10-23.
- [22] S. Ha, I. Rhee, and L. Xu, "CUBIC: A new TCP-friendly high-speed TCP variant," *SIGOPS Oper. Syst. Rev.*, 2008.
- [23] "A comparative analysis of TCP Tahoe, Reno, New-Reno, SACK and Vegas." http://materias.fi.uba.ar/7543/dl/Paper_Congestion_Algorithms.pdf. Accessed: 2017-10-23.
- [24] "TCP congestion control." https://en.wikipedia.org/wiki/TCP_congestion_control. Accessed: 2017-10-23.
- [25] "BIC TCP." https://en.wikipedia.org/wiki/BIC_TCP. Accessed: 2017-10-23.
- [26] "User equipment (UE) application layer data throughput performance," *3GPP TR 37.901 v11.6.1*, 2013.
- [27] S. Bradner and J. McQuaid, "Benchmarking methodology for network interconnect devices," *IETF RFC 2544*, 1999.
- [28] R. Asati, C. Pignataro, F. Calabria, and C. O. Morales, "Device reset characterization," *IETF RFC 6201*, 2011.
- [29] S. Bradner, K. Dubray, J. McQuaid, and A. Morton, "Applicability statement for RFC 2544: Use on production networks considered harmful," *IETF RFC 6815*, 2012.
- [30] "iPerf." <https://iperf.fr/>. Accessed: 2017-03-04.
- [31] "Android tcpdump." <http://www.androidtcpdump.com/>. Accessed: 2017-03-04.
- [32] "Wireshark." <https://www.wireshark.org/>. Accessed: 2017-03-04.
- [33] "Kernel Adiutor." <https://github.com/Grarak/KernelAdiutor>. Accessed: 2017-10-23.
- [34] "D390N LG F60." <http://www.lg.com/at/mobiltelefon/lg-D390N-f60>. Accessed: 2017-03-07.
- [35] "User equipment (UE) radio access capabilities," *3GPP TS 36.306 v14.1.0*, 2016.
- [36] J. Gettys and K. Nichols, "Bufferbloat: Dark buffers in the internet," *Queue*, 2011.

- [37] Y. Xu, Z. Wang, W. K. Leong, and B. Leong, “An end-to-end measurement study of modern cellular data networks,” *Passive and Active Measurement: 15th International Conference, PAM 2014, Los Angeles, CA, USA, March 10-11, 2014, Proceedings*, 2014.
- [38] “RTR-NetTest open data interface documentation.” <https://www.netztest.at/en/OpenDataSpecification.html>. Accessed: 2017-10-31.
- [39] “Spectrogram using short-time Fourier transform – MATLAB spectrogram.” <https://www.mathworks.com/help/signal/ref/spectrogram.html>. Accessed: 2017-05-09.
- [40] A. V. Oppenheim, R. W. Schaffer, and J. R. Buck, *Discrete-Time Signal Processing*. Prentice Hall, 1989.
- [41] L. R. Rabiner and S. R. W., *Digital Processing of Speech Signals*. Prentice Hall, 1978.
- [42] P. Kanuparth and C. Dovrolis, “Shaperprobe: End-to-end detection of isp traffic shaping using active methods,” *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, 2011.
- [43] “Smooth response data – MATLAB smooth.” <https://www.mathworks.com/help/curvefit/smooth.html>. Accessed: 2017-04-20.
- [44] S. Homayouni, V. Raida, and P. Svoboda, “CMPT: A methodology of comparing performance measurement tools,” *2016 8th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, 2016.
- [45] “iPerf – Public iPerf3 servers.” <https://iperf.fr/iperf-servers.php>. Accessed: 2017-03-04.
- [46] “Releases - meefik/busybox - github.” <https://github.com/meefik/busybox/releases>. Accessed: 2017-10-27.
- [47] “Android Studio.” <https://developer.android.com/studio/index.html>. Accessed: 2017-10-26.
- [48] “Android Developers.” <https://developer.android.com/index.html>. Accessed: 2017-10-27.
- [49] “Stack overflow.” <https://stackoverflow.com/>. Accessed: 2017-10-26.
- [50] “Transmitting network data using Volley | Android Developers.” <https://developer.android.com/training/volley/index.html>. Accessed: 2017-10-27.
- [51] H. Holma and A. Toskala, *HSDPA/HSUPA for UMTS: High Speed Radio Access for Mobile Communications*. John Wiley & Sons, 2006.
- [52] H. Holma and A. Toskala, *LTE for UMTS – OFDMA and SC-FDMA Based Radio Access*. John Wiley & Sons, 2009.
- [53] “Apache CouchDB.” <https://couchdb.apache.org/#about>. Accessed: 2017-10-27.
- [54] M. Rindler, P. Svoboda, and M. Rupp, “FLARP, fast lightweight available rate probing: Benchmarking mobile broadband networks,” *2017 IEEE International Conference on Communications (ICC)*, 2017.

- [55] S. Homayouni, V. Raida, P. Svoboda, and M. Rupp, “The impact of duration and settings of tcp measurements on available bandwidth estimation in mobile networks,” *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, Montreal*, 2017.
- [56] “RTR-NetTest.” <https://www.netztest.at/en/>. Accessed: 2017-03-09.
- [57] “Open-RMBT.” <https://github.com/alladin-IT/open-rmbt>. Accessed: 2017-03-09.
- [58] “RMBT: Specification.” <https://www.netztest.at/doc/>. Accessed: 2017-03-09.
- [59] “AKOSTest.” <https://www.akostest.net/en/>. Accessed: 2017-10-27.
- [60] “AKOSTest – Open-Data.” <https://www.akostest.net/en/opendata>. Accessed: 2017-10-27.
- [61] “Merač internetu.” <https://www.meracinternetu.sk/en/about>. Accessed: 2017-10-27.
- [62] “Merač internetu | open data.” <https://www.meracinternetu.sk/en/opendata>. Accessed: 2017-10-27.
- [63] “RATEL NetTest.” <https://www.nettest.ratel.rs/en/about>. Accessed: 2017-10-27.
- [64] A. P. Foong, T. R. Huff, H. H. Hum, J. R. Patwardhan, and G. J. Regnier, “TCP performance re-visited,” *2003 IEEE International Symposium on Performance Analysis of Systems and Software. ISPASS 2003.*, 2003.
- [65] B. Wun and P. Crowley, “Network i/o acceleration in heterogeneous multicore processors,” *14th IEEE Symposium on High-Performance Interconnects (HOTI'06)*, 2006.
- [66] “The drawbacks of local packet captures | packet foo.” <http://blog.packet-foo.com/2014/05/the-drawbacks-of-local-packet-captures/>. Accessed: 2017-11-07.
- [67] “TrafficStats | Android Developers.” <https://developer.android.com/reference/android/net/TrafficStats.html>. Accessed: 2017-04-18.
- [68] “ASU Wert – Signalstaerke messen und interpretieren.” <http://www.lte-anbieter.info/technik/asu.php>. Accessed: 2017-11-09.
- [69] C. Midoglu, L. Wimmer, and P. Svoboda, “Server link load modeling and request scheduling for crowdsourcing-based benchmarking systems,” *2016 International Wireless Communications and Mobile Computing Conference (IWCMC)*, 2016.
- [70] “RTR-NetTest | Statistics.” <https://www.netztest.at/en/Statistik>. Accessed: 2017-03-09.