



Measurement of Position-based Network-Characteristics in Cellular Networks while Moving

Master's Thesis

by

Tobias Krauthoff

born in

Wesel

submitted to

Professorship for Computer Networks and Communication System

Prof. Dr. Martin Mauve

Heinrich-Heine-Universität Düsseldorf

November 2014

Supervisor:

Norbert Goebel, M. Sc.

Abstract

In this thesis a novel algorithm for logging data rate, drop rate and latency in mobile cellular networks is introduced. The algorithm is able to cope with changing network characteristics due to fallbacks, handovers and reduces self-induced congestion between chirps to a minimum.

Before this thesis was commenced, measurements in mobile cellular networks while moving were distorted through inter-chirp self-induced congestion. This congestion leads to a distorted measurement of packet latencies. Hence, in the context of this thesis new algorithms for handling changing network characteristics have been developed. This also includes detection of congestion within a packet train, between packet trains as well as a clever mechanism for sending notifications based on congestion. Therefore a server and a testbed were set up. The server acts as counterpart for the mobile unit and the testbed serves for controlled measurements while developing. Important for the measurement is, that the structure of the cellular network is seen as black box.

The measurement algorithm has been extensively tested under laboratory conditions as well as at a stationary point and while moving. For the wired testbed with laboratory conditions, cross-traffic generators were evaluated. Measurements in the real world have been done with fixed positions as well as while moving. The algorithm was tested with numerous measurements and it was shown that the algorithm cope with changing network characteristics. Therefore self-induced inter-chirp congestion can be decreased to a minimum.

Acknowledgments

Lots of people supported me during my work on this thesis to whom I express my gratitude.

At first there is my girlfriend, who kept me on track all the time. Especially in times of exhausting and time consuming work or during measurements rides in the early morning. She was always there for me and supported me.

I would also like to thank all my colleagues and friends, who have helped out with advice, assistance and especially linguistic corrections. This especially applies to my supervisor, who challenged my work constructively during many hours of discussion.

Contents

List of Figures	xi
List of Tables	xiii
List of Listings	xv
Nomenclature	xvii
1 Introduction	1
1.1 Structure of this Thesis	2
2 Related Work	3
2.1 Models for Available Data Rate Measurements	3
2.1.1 PGM – Probe Gap Model	4
2.1.2 PRM – Probe Rate Model	4
2.2 Bulk Traffic	5
2.3 Trace-based UMTS Simulation	6
3 Fundamentals	7
3.1 Network Characteristics in Cellular Mobile	7
3.2 RMF – Rate Measurement Framework	10
3.2.1 Channels	11
3.2.2 Threads	13
3.3 Synchronization and Positioning Systems	15
3.3.1 NTP – Network Time Protocol	15
3.3.2 GPS – Global Positioning System	16
3.3.3 Galileo, GLONASS and others	17
3.4 Hardware for Mobile Measurements	17
3.5 Traffic Generators	18

4	Framework Improvements	21
4.1	Posix Thread Wrapper	21
4.2	GPS Helper Class	21
4.3	NTP Helper Class	22
4.4	Result Calculator	24
4.5	Countdown-Timer	25
4.6	SICD – Self-Induced-Congestion Detector	25
5	Novel Measurement Algorithm	27
5.1	Assumptions	27
5.2	Starting a Measurement	28
5.3	Sending packets via measurement- and safeguard-channel	29
5.4	Receiving Packets	30
5.5	SIC – Inter-Chirp Self-Induced Congestion	31
5.5.1	Self-Induced Congestion within current Chirp	31
5.5.2	Self-Induced Congestion between Chirps	32
5.5.3	Linear Regression	35
5.5.4	PCT – Pairwise Comparison Test and PDT – Pairwise Difference Test . .	36
5.5.5	Calculation of transmit time of the next chirp	37
5.5.6	Positive and Negative Feedback	42
6	Experiments	45
6.1	Measurements with Laboratory Conditions	45
6.1.1	Hardware	45
6.1.2	Evaluation of Traffic Generators	46
6.1.3	RMF Input Parameters	49
6.1.4	Estimating Chirp Length and Packet Size	50
6.1.5	Measurements	53
6.2	Real Life Measurements using Mobile Cellular Networks	62
6.2.1	Analyzing Data with KML-Files	63
6.2.2	Mobile cellular network measurement using a stationary client	64
6.2.3	Mobile cellular network measurement using a moving client	67
7	Summary and Outlook	73
7.1	Summary	73
7.2	Outlook	74

A	Testbed	75
A.1	Maximal data rate	75
A.2	Packet Loss while using Wondershaper	76
A.3	Scripts	78
A.3.1	CPU Frequency	78
A.3.2	Cross-Traffic	78
B	Rate Measurement Framework v. 58894	81
B.1	Interfaces	81
B.2	Parameters	81
C	Configuration Files	83
C.1	UMTS Login	83
C.2	gpsd configurations	85
C.3	NTP configurations	85
C.4	ARP-Reply's and Source-Based-Routing	88
D	Miscellaneous	91
D.1	GPSD Data	91
D.2	Gpspipe Data	92
D.3	NTPQ Data	92
	Bibliography	93

List of Figures

2.1	Representation of the Probe Gap Model	4
2.2	Example sending data rate of an chirp using the Probe Rate Model	5
3.1	Inter-Cell Handover in cellular networks	8
3.2	Initialization of a measurement with the RMF	11
3.3	Channels of communication of the RMF	12
3.4	Threads, queues and helper classes of the RMF	15
4.1	NTP offset from PPS and time servers of the university for August 2014	24
4.2	NTP offset from PPS and time servers of the university for September 2014	25
5.1	Presenting time stamps within a chirp	31
5.2	Detecting self-induced congestion within current chirp	33
5.3	Detecting self-induced congestion between chirps	34
5.4	Values of PCT and PDT for measurement in Figure 6.13	37
5.5	Presentation of “time to sleep”	37
5.6	Importance of the gap time	39
5.7	Flow chart of the logic after detecting self-induced congestion	41
5.8	Logic of the feedback-mechanism for the sleep time on congestion	43
6.1	Testbed for measurements	47
6.2	Cross-traffic by different generators	48
6.3	Data rate dependent on different packet sizes	51
6.4	Data rate dependent on different chirp lengths	52
6.5	Calculation of packet size and chirp length	52
6.6	Measurement results for an unencumbered link	54
6.7	Measurement results for a link with loss	55
6.8	Measurement results with one cross-traffic flow, without congestion	56
6.9	Measurement results with three cross-traffic flows, without congestion	56
6.10	Measurement results with two cross-traffic flows and with congestion	57

6.11	Measurement results with two cross-traffic flows and several congestion periods .	58
6.12	Measurement results with one random cross-traffic flow	59
6.13	Measurement results and delays with two random cross-traffic flows	60
6.14	CDF of gap time and backbone delay for the upload of the client in Figure 6.13 .	61
6.15	Example of a measurement ride in Google Earth	63
6.16	Data rates of the client at October 29th	64
6.17	Delays and gap times of the measurement at October 29th, 11:15pm, without SICD	65
6.18	Delays and gap times of the measurement at October 29th, 11:17m, with SICD .	65
6.19	Delay and backbone delays of the client at October 29th	66
6.20	Delay of the first packet in chirps of the client at October 31th	67
6.21	Route of the measurement drive shown in Google Earth	68
6.22	Values of the client for the measurement ride on November 4th, 10:55pm	68
6.23	CDF for delays of the measurement ride on November 4th, 10:55pm	69
6.24	CDF for gaps of the measurement ride on November 4th, 10:55pm	69
6.25	Values of the client for the measurement ride on November 4th, 11:05pm	70
6.26	CDF for delays of the measurement ride on November 4th, 11:05pm	70
6.27	CDF for gaps of the measurement ride on November 4th, 11:05pm, with SICD .	71

List of Tables

3.1	Data rates in mobile cellular networks	9
3.2	Format of the measurement packets	13
3.3	Shortcuts used in log files	14
5.1	Payload data of the measurement packet	28
5.2	Dynamic payload data of the safeguard packets	30
5.3	Bitmask for the send reason in the payload of safeguard packets	30
5.4	Acronyms for time stamps and packet properties	32
6.1	Hardware of the testbed	46
6.2	Data rates of cross-traffic for Brute and D-ITG dependent on time	48
6.3	Input parameters for the new measurement method	49
6.4	Packet sizes in dependent of time	50
6.5	Data rate in dependent of time	51
6.6	Options for measurements in the testbed	53
6.7	Periods of loss for testing	54
6.8	Technical data for the server and mobile node	62
6.9	Parameters for the server and client for real life measurements	63
6.10	Statistics on different measurement in real life	72
B.1	RMF interfaces for a measurement algorithm	81
B.2	Local settings for the RMF	82
B.3	Local settings for the server and client	82
B.4	Global settings for the RMF	82

List of Listings

A.1	Measuring bandwidth between Amy and Fry: client side	75
A.2	Measuring bandwidth between Amy and Fry: server side	76
A.3	Measuring bandwidth between Amy and Fry with shaped router: client side	76
A.4	Measuring bandwidth between Amy and Fry with shaped router: server side . . .	77
A.5	/etc/default/cpufrequtils	78
A.6	Cross-traffic script for brute	78
A.7	Cross-traffic script for D-ITG	79
A.8	Cross-traffic script for IPerf	79
C.1	Client: /etc/wvdial.fonic.conf	83
C.2	Client: /etc/wvdial.o2.conf	84
C.3	Client: /etc/wvdial.tmobile.conf	84
C.4	Additional wvdial commands	85
C.5	Client: /etc/default/gpsd	85
C.6	Server: /etc/ntp.conf	85
C.7	Client: /etc/ntp.conf	86
C.8	Server: /etc/cron.daily/ntp	87
C.9	Server: /etc/sysctl.conf	88
C.10	Server: /etc/network/interfaces	88
C.11	Server: /etc/iproute2/rt_tables	89
C.12	Server: /etc/rc.local	89
C.13	Server: ip route show	89
C.14	Server: ip route show table eth0	89
C.15	Server: ip route show table eth1	89
D.1	Data from the gps receiver	91
D.2	First lines out of the log from gpspipe	92
D.3	ntpq -p	92

Nomenclature

List of Abbreviations

3GPP	Third Generation Partnership Project	7
ARP	Address Resolution Protocol	23
BSC	Base-Station Controller	8
C2C	Car-to-Car Communication	1
C2I	Car-to-I Communication	1
C2X	Car-to-X Communication	1
CBR	Constant Bit Rate	18
CDF	Cumulative distribution function	61
CDTI	Countdown Timer	14
CRUDE	Collector for RUDE	19
D-ITG	Distributed Internet Traffic Generator	19
EDGE	Enhanced Data Rates for GSM Evolution	9
GNSS	Global Navigation Satellite System	17
GPRS	Global Packet Radio Service	9
gpsd	GPS Daemon	21
GPSH	Global Position System Helper	14
GSM	Global System for Mobile Communications	9

List of Listings

HSCSD	High Speed Circuit Switched Data	9
HSDPA	High Speed Download Packet Access	10
HSPA+	High Speed Packet Access Plus	10
HSUPA	High Speed Upload Packet Access	10
IRNSS	Indian Regional Navigation Satellite System	17
JVM	Java Virtual Machine	11
LAN	Local Area Network	1
LTE	Long Term Evolution	10
LTE-A	Long Term Evolution Advanced	10
MAIN	Main Program	14
MEME	Measurement Methods	14
MGEN	Multi-Generator	19
MIMO	Multiple In Multiple Out	10
NAT	Network Address Translation	11
NAVSTAR	Navigational Satellite Timing and Ranging	15
NDA	Non-Disclosure Agreement	1
NGC2X	Next Generation Car-2-X Communication	1
NTP	Network Time Protocol	15
NTPH	Network Time Protocol Helper	14
OWD	one-way delay	36
PCT	Pairwise Comparison Test	33
PDT	Pairwise Difference Test	33
PGM	Probe Gape Model	4
PID	program identity number	22

PPS	Pulse-Per-Second	16
PRM	Probe Rate Model	4
QZSS	Quasi-Zenith Satellite System	17
RECA	Result Calculator	14
RNC	Radio-Network Controller	8
RTT	Round Trip Time	5
RUDE	Real-time UDP Data Emitter	19
SIC	Self Induced Congestion	4
SICD	Self Induced Congestion Detector	14
SIM-TD	Sichere Intelligente Mobilität - Testfeld Deutschland	1
TBUS	Trace-based UMTS Simulation	6
TCP	Transmission Control Protocol	16
TMME	Measurement Receiver	14
TMSE	Measurement Sender	14
TPV	Time-Position-Velocity	22
TRDD	Measurement Receiver Logging	14
TSDD	Measurement Sender Logging	14
TSRE	Safeguard Receiver	14
TSSE	Safeguard Sender	14
UDP	User Datagram Protocol	11
UMTS	Universal Mobile Telecommunications System	10

List of Symbols

$d_i^{air}(j)$	Approximated air delay of the j th packet in chirp i	38
d_{owd}	One-way delay of the last non-congested packet	40

List of Listings

Gbit/s	Giga bit per second, 10^9 Bit/s	9
KByte/s	Kilo byte per second, 10^3 Byte/s	5
Kbit/s	Kilo bit per second, 10^3 Bit/s	9
Mbit/s	Mega bit per second, 10^6 Bit/s	9
t^{ct}	Time of the countdown-timer	42
t_i^{gap}	Gap time between chirp i and $i + 1$	39
$t_i^{tx}(k)$	Transmit time stamp of packet k of chirp i	32
A_i	Calculated data rate for chirp i	32
$b_f(1)$	First byte of the first packet in chirp i	39
$b_l(n_i)$	Last byte of the last packet in a chirp i	39
d_{bb}	Backbone delay of the last non-congested packet	38
n_i	Count of packets of chirp i	32
S_{PCT}	Factor of the pairwise comparison test [RRBLC03]	36
S_{PDT}	Factor of the pairwise difference test [RRBLC03]	36
sp_i	Packet size of packets $2, \dots, n_i$ of chirp i in Bytes	32

Chapter 1

Introduction

The development of electronic technology had large influences on the society in the last decades. One of the biggest innovations is mobile internet. Even automobiles are connected to the internet for purposes like entertainment, navigation or to increase safety. Unsurprisingly, science and industry are interested in communication between cars. Their goal is to improve the comfort of travelling by car as well as enhance systems that provide safety to the passengers. The whole area is called *Car-to-X Communication* (C2X) and is divided into communication between cars, *Car-to-Car Communication* (C2C) and communication with infrastructure like *Road Side Units*, so called *Car-to-Infrastructure Communication* (C2I).

Hence, the professorship for computer networks and communication system at the Heinrich-Heine-Universität Düsseldorf has interest on this topic, too. They participated in projects like “Next Generation Car-2-X Communication (NGC2X)” and “Sichere Intelligente Mobilität - Testfeld Deutschland (SIM-TD)”. Both projects have in common, that explorations and developments have to be tested in field tests, which are very costly. Therefore, simulations are used. Most of them are based on Wireless LAN instead of cellular networks. However, simulations may be different when cellular networks are used. Though many simulators for cellular networks are commercial and based on a *non-disclosure agreement* (NDA). Due to this agreement, these simulators are not suitable for scientific applications. NDAs hold, because information about the cellular networks are not discretionary. This is the reason, why a rate measurement framework was acquired in a previous thesis. Nevertheless, congestion between packet trains while measuring is still a problem in cellular networks. Hence, the aim of this thesis is to develop a novel algorithm for measurements of position-based network-characteristics in cellular networks while moving. The aspect of a moving measurement device is highly important, however tracking of trace-based position is not meant. Movements in cellular networks cause changes of network characteristics.

As consequences fallbacks, handovers and self-induced congestion between packet trains could occur. The problem of congestion was not solved satisfactorily yet.

1.1 Structure of this Thesis

The structure of this thesis is explained briefly below:

- Chapter 2 introduces related work. First a trace-based simulation program will be described. This is completed by general models are explained as well as the model which is used in this thesis.
- Chapter 3 briefly describes fundamentals in cellular networks. Additionally, a short introduction into the network time protocol (NTP) and global positioning system (GPS) is given as well as their usage for measuring data rates in cellular networks. Further terminology in those areas will be discussed and the rate measurement framework will be explained briefly. Lastly different traffic generators for the testbed are illustrated.
- Chapter 4 furtheron describes improvements, additions and novelties for the rate measurement framework.
- Chapter 5 explains the novel algorithm for measurements of delay, available data rate and loss rate. Furthermore design and implementation of the measurement algorithm for the rate measurement framework are introduced. This includes models for detecting self-induced congestion as well as the mechanism for sending feedback.
- Chapter 6 deals with evaluations and analyses of different tests under labour and real-life conditions, which were made to improve and test the new algorithm. These tests were done in a wired testbed under laboratory conditions as well as in real-life with fixed positions and while moving.
- Last but not least chapter 7 contains a summary and an outlook.

Chapter 2

Related Work

This chapter briefly introduces related work. Section 2.1 describes two paradigms for bandwidth measurements, while Section 2.2 describes the model, which is used for estimating the available data rate¹ in the new measurement algorithm. Section 2.3 further explains the *trace-based UMTS simulation* (TBUS) and summarizes a paper, which is based on TBUS.

2.1 Models for Available Data Rate Measurements

End-to-end network measurements are used to estimate bandwidth of given network paths from one terminal to another. But instead of full utilization – like a download does – these measurements do not measure bandwidth continuously. They only measure the available data rate for a specific moment, for example half a second every second. Therefore, there is a trade-off between accuracy and utilization. Moreover, current and well established measurement methods were developed for 802.3 Ethernet environments, not for cellular networks. Hence, sharp drops in the data rate are not considered and reveal problems. Current and available end-to-end measurement methods can be divided into two paradigms:

1. The *probe gap model* (PGM) will be explained in Section 2.1.1 and
2. the *probe rate model* (PRM) will be described in Section 2.1.2.

Both paradigms either use *pairs of packets* or *trains of packets*, which are called *chirps*. A chirp is a sequence of measurement packets with varying gaps between adjacent packets. In summary the paradigms and other measurement methods differ in the ways packets are sent and analyzed.

¹The terms *bandwidth* and *data rate* are used interchangeably in this thesis.

2.1.1 PGM – Probe Gap Model

The PGM uses *packet dispersion* for calculating the available bandwidth. Packet dispersion is the time lag between two adjacent packets at the receiver, whereby the sender sent these packets back-to-back. Gaps can occur due to foreign traffic respectively *cross-traffic*. The path of cross traffic is partly equal to the measurement traffic through the network. The functional principle of PGM can be seen in Figure 2.1: Two or more packets are sent within the time of Δ_{in} and reach the receiver within the time of Δ_{out} . The available bandwidth A is:

$$A = C \left(1 - \frac{\Delta_{out} - \Delta_{in}}{\Delta_{in}} \right)$$

It takes $\Delta_{out} - \Delta_{in}$ time to transmit the cross-traffic, which has the rate of $C \cdot \frac{\Delta_{out} - \Delta_{in}}{\Delta_{in}}$, where C is the capacity of the bottleneck. Therefore, $1 - \frac{\Delta_{out} - \Delta_{in}}{\Delta_{in}}$ parts of the channels bandwidth is used for the measurement packets. This method is used in Delphi, IGI and Spruce (see [RCR⁺00], [HS03] and [LB90]).

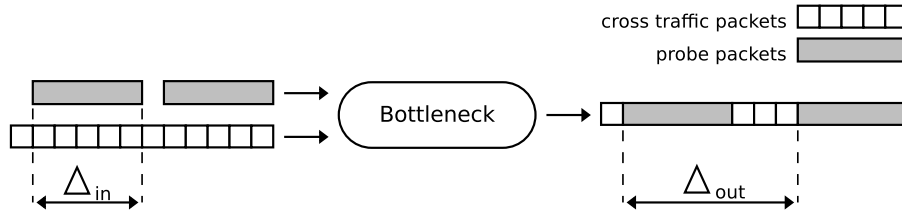


Figure 2.1: Representation of the Probe Gap Model (based on [LB90])

While PGM theoretically just needs two packets to calculate the available data rate, it has some drawbacks. First, only two packets are needed and cover a very short time interval, therefore there is a high variance time. Second this model assumes *first-in-first-out* queues, which are not always given. Third the *tight link* and the *narrow link* are the same. Tight link is the latest router with the smallest available bandwidth and narrow link denotes the router with the smallest capacity on an network path. When the cross-traffic has another route than the measurement packets due to different routes, the bandwidth could be underestimated (see [LDS06], [Goe10], [LB90] and [GKMK14]).

2.1.2 PRM – Probe Rate Model

The PRM uses the principle of *self-induced congestion* (SIC) for estimating the available bandwidth for the current network path. If the sender transmits measurement packets at a rate lower

than the recent available data rate along the network path, then the downlinks' data rate of the receiver is equivalent to the sending data rate. In contrast, if the sender transmits measurement packets at a rate higher than the recent available data rate along the network path, then queues inside the network will build up and SIC will occur. This model is shown in Figure 2.2. First a small initial sending rate of 100 KByte/s is shown. Then the data rate is increased up to 1500 KByte/s until SIC occurs since 1250 KByte/s. As a consequence the receiving data rate will be lower than the sending data rate. If the sender does not get to know about the congestion, it will continue sending and the delay measurement will be distorted. Hence, congestion has to be reduced, because time stamps are highly important for calculating the current available data rate. PRM is used in Assolo, Pathload, Pathchirp, PTR and TOPP (see [GRT09], [JD02], [RRBLC03], [LB90] and [LB90]).

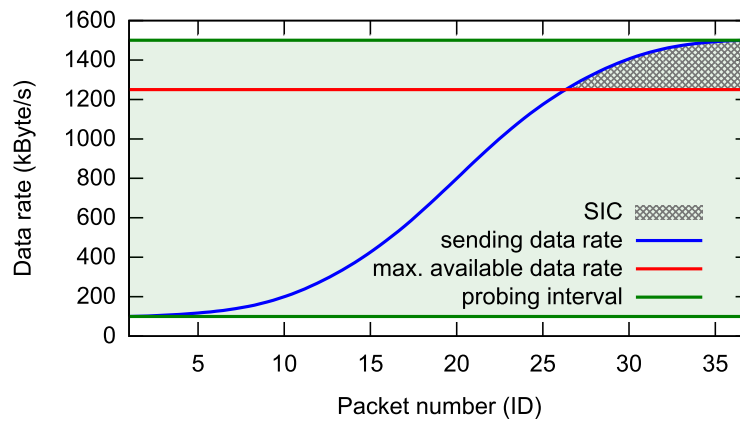


Figure 2.2: Example sending data rate of a chirp using the Probe Rate Model

PRM has the advantage, that the available data rate can be estimated accurately. But for this the cross-traffic has to be constant and multiple RTTs are needed. Though the sender is moving while measuring in cellular networks and therefore he is not able to repeat a measurement at the exact same place with the same influences. Additionally, the cross-traffic is not constant, because the sender is not the only one in the mobile cell. Therefore, PRM is not suitable for measurements in cellular networks while moving.

2.2 Bulk Traffic

Due to the properties of PGM and PRM (see 2.1.1 and 2.1.2) those are not suitable for measurement in cellular networks while moving. Therefore [Goe10] used bulk traffic for measurements

in cellular networks. Bulk traffic consists of many chirps, whereby the packets in the chirp are sent back-to-back. Therefore, the data rate A can be estimated with formula (see [Lan13, page 23]):

$$A = \frac{(N-1) \cdot L}{\Delta_R}.$$

L is the packet size, Δ_R is the chirp dispersion respectively the delay of transmission. [Goe10] enhanced this method, by piggybacking measurement data in the measurement packets. The complete algorithm will be presented step-by-step in Chapter 5, in which bulk traffic is relevant for detecting self-induced congestion in Section 5.5.

2.3 Trace-based UMTS Simulation

TBUS contains a logger and an analyzer (see [Goe10]). The logger is responsible for saving GPS data, information about the UMTS modem as well as delay, drop and data rates of chirps. The analyzer processes saved data for plotting, for example with gnuplot, a command-line driven graphing utility (see [WK14]). All data is generated through real-world measurements, where measurements were done between a stationary and a mobile node. During these rides measurement packets are sent from a mobile client to a server via cellular network, which is connected to the internet. In [Goe10] focus was on available bandwidth, latency and drop rate. Information, which would be based on the internal structure of the cellular network, are not used and are seen as a black box. One result of [Goe10] is, that recent data rate measurements are not suitable for cellular networks, as explained in Section 2.1.

Later the work about TBUS was intensified and [GKMK14] published. Furthermore, it turned out that the measurement method was insufficient and there is still room left for improvements. Especially self-induced congestion could be reduced. The paper describes a trace-based simulation model derived from real-world measurement. The simulation model is based on bulk traffic (see Section 2.2), because other models are not suitable (see Section 2.1.1). Focus was on available bandwidth, latency and the drop rate, too, but there are many improvements in analyzing the data. Therefore, SIC can be detected while analyzing the data, but not prevented while measuring. Hence, one main problem is SIC. This thesis presents a model providing resolutions to reduce self-induced congestion.

Chapter 3

Fundamentals

In this chapter fundamentals about cellular networks (see Section 3.1) and the Rate Measurement Framework (see Section 3.2) are explained. The former is about terminology and a brief insight into current network characteristics. The latter deals with a special framework for data rate measurements in cellular networks, which was introduced in a master thesis' by [Wil12]. Moreover, the third section contains a subsection about time synchronization and positioning systems. Section 3.4 is about the hardware, which is used for the mobile node and the last section is with an introduction of traffic generators, which are used in the testbed.

3.1 Network Characteristics in Cellular Mobile

On designing a measurement method for cellular networks, an occasion to the standards of cellular networks must be given. Therefore, a short explanations of the terms *handover* and *fallback* are necessary. Both events could trigger SIC and should be considered when developing a measurement method. Standards in cellular networks are passed by the *Third Generation Partnership Project*, or short 3GPP (see [3GP14a]). The standards and data rates are important for measuring, because they have a strong influence on the algorithm in Section 6.1.3. The mobile cellular network was designed as circuit-switched one, but recent developments and trends moved it to a packet-switched network. However, new techniques offer new problems – these include handovers and fallbacks.

Handover During a handover the controller of a mobile end device changes (see [3GP09], [Sch03]). The controller is a network element in mobile networks for – amongst other things

– controlling the connection of the end device. Dependent on the network, the controller can be a base-station controller (BSC) or radio-network controller (RNC). For example the BSC is responsible for all wireless connections in the GSM-network and prompts power controls as well as handovers. The detailed procedure is explained in [3GP09, page 26]. This change between controllers has to be done, because the signal strength of the old controller is too weak and there is another controller with a better signal. An illustration of an inter-cell handover can be seen in Figure 3.1. Inter-cell handover is the procedure, when the mobile radio cell will be changed, but the controller stays the same.

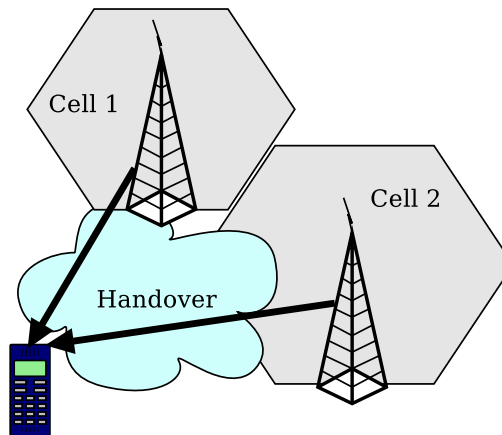


Figure 3.1: Inter-Cell Handover in cellular networks

According to a statement made by [3GP09] there are four kinds of handover-procedures, but they differ only in the participation of different network elements. Every handover has in common, that a change of the controller causes new network characteristics. This affects the route, data rate and latency. Additionally it is possible, that the up- and downlink are using different protocols, which leads to different maximum data rates for up- and downstream. Table 3.1 gives a short overview of some of the used transmission technologies, whereby the data rates are divided into down- and uplink.

Fallback When a fallback occurred, the current data rate decreases significantly (see [ETS14], [Sch03]). This affects the current measurement, because then the terminals assume a data rate, which is too high. Hence, SIC is possible and this has to be recognized and treated.

Data rates in mobile cellular networks Table 3.1 shows different data rates for three generations of mobile cellular networks (see [Sch03], [Lan13] and [Gra13]). The specified values are not exact, because for example the available data rates are depending on the level of the system

components or class of the terminal, which are selected by the provider. Data rates listed in Table 3.1 are most commonly used.

Standard	Technology	Data rate downstream	Data rate upstream
2G	GSM	14.4 Kbit/s	14.4 Kbit/s
	HSCSD	57.6 Kbit/s	28.8 Kbit/s
	GPRS	53.6 Kbit/s	26.8 Kbit/s
	EDGE	236.8 Kbit/s	118.4 Kbit/s
3G	UMTS	384 Kbit/s	128 Kbit/s
	HSDPA / HSUPA	7.2 Mbit/s	5.8 Mbit/s
	HSPA+	42.2 Mbit/s	11.5 Mbit/s
4G	LTE	300 Mbit/s	150 Mbit/s
	LTE-A	3 Gbit/s	1.5 Gbit/s

Table 3.1: Data rates in mobile cellular networks

2G – GSM / HSCSD / GPRS / EDGE The second generation (2G) of the mobile telephone system is divided into four standards (see also [Sch03] and [Gra13]) and the corresponding data rates can be seen in Table 3.1:

- *Global System for Mobile Communications* (GSM)
- *High Speed Circuit Switched Data* (HSCSD)
- *Global Packet Radio Service* (GPRS)
- *Enhanced Data Rates for GSM Evolution* (EDGE)

GSM is the first standard which uses a digital transfer of the data. While using GSM a mobile terminal is assigned to different time slots. The data rates offered by GSM are up to 14.4 Kbit/s, first line in Table 3.1. HSCSD allows the terminal to combine multiple time slots of GSM therefore data rates of up to 57.6 Kbit/s are possible for the downstream and 28.8 Kbit/s for the upstream. The third technology is GPRS. Compared to the circuit-switched technology GSM and HSCSD, GPRS offers packet-switched data service. Hence, in dependence of the chosen error correction code and count of used time slots download rates up to 53.6 Kbit/s and uploads of 26.8 Kbit/s are possible (see [3GP14b]). Last but not least, there is EDGE as extension for GSM. EDGE uses a new modulation scheme compared to GSM, so as a function of the terminal class data rates up to 236.8 Kbit/s are possible (see [Gra13], [3GP14b]).

3G – UMTS / HSDPA/HSUPA / HSPA+ The third generation (3G) is divided in three standards, which were released over the years by [3GP14a] (see also [Sch03], [Gra13]) and the respective data rates can be seen in Table 3.1, too:

- *Universal Mobile Telecommunications System (UMTS)*
- *High Speed Download Packet Access (HSDPA)* and
High Speed Upload Packet Access (HSUPA)
- *High Speed Packet Access Plus (HSPA+)*

UMTS is the first standard in the third generation of cellular networks (see [3GP14f]) and the data rates can be seen in the fifth row of Table 3.1. Compared to 2G the radio interface was revised, resulting in new technologies providing increased data rates. Finally, there is HSPA (see [3GP14c]), which is a collective for extension of the UMTS standard and HSPA+ as extension of HSPA. HSPA can be divided in down- and upload, so the protocols for the different links are called HSDPA and HSUPA. For higher data rates a new modulation scheme and multiple antennas (called *Multiple In Multiple Out* – MIMO) are used. Even HSPA could be improved with dual carrier, respectively dual cell technology, and is called HSPA+ then. This technique bunches two 5-MHz channels to achieve higher data rates.

4G – LTE / LTE Advanced The fourth generation (4G) of the mobile telephone system is currently divided into two standards (see also [Sch03], [Gra13]) and the corresponding data rates can be seen in Table 3.1:

- *Long Term Evolution (LTE)*
- *Long Term Evolution Advanced (LTE-A)*

Both are the next standard and generation of cellular networks. LTE offers data rates up to 300Mbit/s in the down- and 150Mbit/s in the uplink (see [3GP14d]). LTE Advanced increases the available data rates up to 3Gbit/s in the down- and 1.5Gbit/s in the uplink for (see [3GP14e]). All four values are the highest theoretical peak data rates on the transport channel.

3.2 RMF – Rate Measurement Framework

The Rate Measurement Framework was developed by [Wil12] and enhanced by [Lan13]. It is written in C++, therefore the system governor or scheduling policies could be manipulated fast

and easy. For example, in Java every access is done in the *Java Virtual Machine* (JVM), thus there would arise a further delay. The basic idea of the RMF is that the developer of a process for mobile measurement only has to implement methods for filling the sending queue and estimating the current data rate. The framework itself will handle sending and receiving packets (see Tabular B.1). Hereafter the implemented channels – see Section 3.2.1 – and the architecture of all threads – see Section 3.2.2 – are explained.

3.2.1 Channels

The RMF provides methods to execute end-to-end measurements, which require two terminals as the respective ending points. All measurements are done between a mobile client node and a stationary server. [Goe10] describes, that mobile network providers often use NAT (Network Address Translation) in their core network, therefore the mobile devices are not directly accessible in the network. Therefore, the server has to be available for a certain combination of IP and port number for UDP and TCP packets.

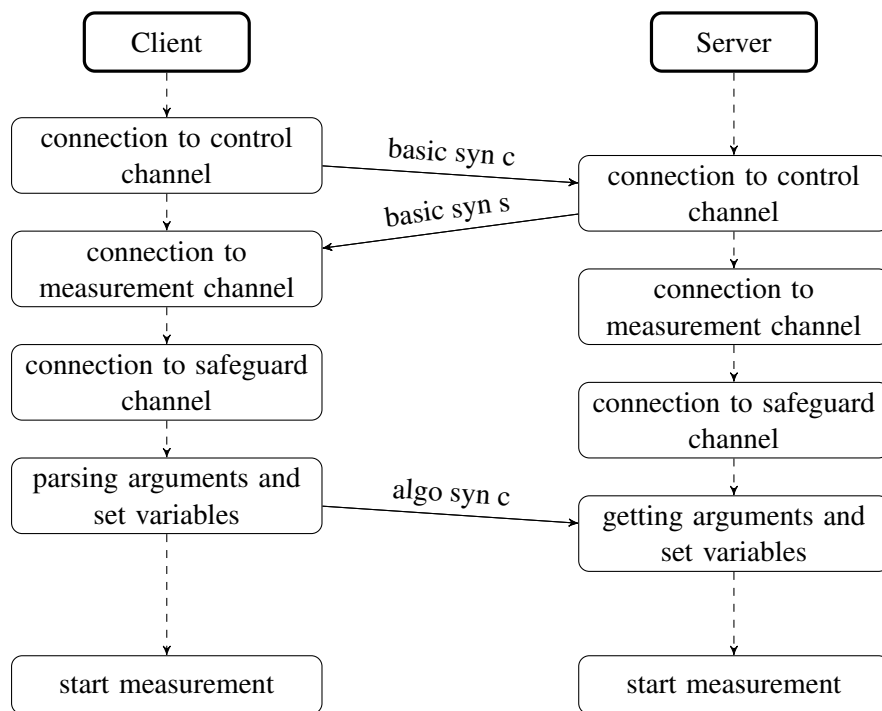


Figure 3.2: Initialization of a measurement with the RMF

Figure 3.2 depicts the communication between client and server, when starting a measurement. First the global input parameters of the users input will be parsed and the control channel will

be bound. The global part contains parameters with basic information – they are listed in Table B.4. Table B.3 is about settings for server and client. Next to these parameters, local settings for each terminal can be set. They are listed in Table B.2. Then a message for synchronizing client and server will be sent from the client to the server via control channel (`basic syn c`). On receiving, the server answers with further information (`basic syn s`) and binds his measurement channel. When the client receives the servers response, it will connect the safeguard channel, when the parameter was set. The server binds its safeguard channel after receiving the first synchronization message, because in this synchronization message the necessary information are included. After this the client parses the second part of the input, which contains specific information for the used measurement method and sends a last message for synchronizing (`algo syn c`). The parameters can be seen in Table 6.3. Finally, the measurement will be started.

For distinguishing between the channels, they will be explained briefly. Figure 3.3 depicts the communication between the client and server again, which is based on two or three real and one virtual channel. The third real channel is based on the use of an safeguard channel. The *control channel* maintains the connection establishment, hence it is realized with TCP. Every connection establishment contains essential parameters for measuring (see Table 6.3, B.2 and B.4). If conflicts occur, which were induced through the choice of parameters, the client will have a higher priority and overrules the server. If a termination of the connection is noticed, the client will stop current measurements and will terminate itself. By contrast the server only terminates the measurement and restarts the RMF to be available for new measurements.

The *measurement channel* is used for the measurement traffic. This channel is implemented as a UDP-connection, because repeated transmission of the same packets would disturb calculations in the measurement algorithm. With the use of UDP, packets could change their order or get dropped. This has to be considered while developing.

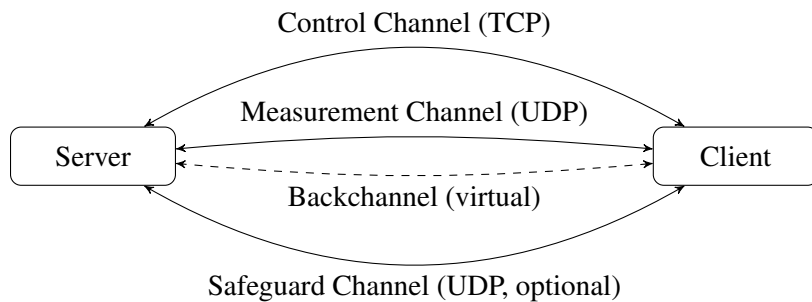


Figure 3.3: Channels of communication of the RMF (based on [Lan13, page 8])

The third channel in Figure 3.3 is the *backchannel*. It bases on the measurement channel, because

all information of the backchannel is piggybacked in the payload of measurement packets in the measurement channel. Therefore, the payload is divided in a fixed and a dynamic part. The fixed part is determined when the packet is given to the sending thread, while the dynamic part is registered before sending (see [Lan13]). The backchannel uses the dynamic part. Format of the measurement packet shown in Table 3.2, whereby the payload will be described in Section 5.2.

Field	Content	Length
IP-Header	IPv4 / IPv6 Address	20 / 40 Byte
UDP-Header	Port-Addresses	8 Byte
Sendtimestamp integer part	0 - 4.294.967.295	4 Byte
Sendtimestamp fraction	0 - 1.000.000.000	4 Byte
Fixed Payload Length	1 - 65535	4 Byte
Fixed Payload Content	–	1 Byte x Fixed Payload Length
Dynamic Payload Length	1 - 65535	4 Byte
Dynamic Payload Content	–	1 Byte x Dynamic Payload Length

Table 3.2: Format of the measurement packets, based on [Wil12, page 35]

The *safeguard channel* is the third real channel. It is necessary, because the measurement channel could be affected by SIC. Therefore, the safeguard channel, which is a UDP-connection, should transfer packets in the case of inter-chirp congestions². While measuring in cellular networks, the safeguard channel could be the network of a different provider, which is not stressed through measurements. A more detailed view will be given in Chapter 5.

3.2.2 Threads

The RMF is divided into many threads for easier handling and performance improvements. Especially a multi-core system increases the performance, because threads can run in parallel through multiple cores. An overview of all classes is given in Table 3.3. This table also shows, whether the class is implemented as a thread or not. The architecture itself can be seen in Figure 3.4. Class one until eight in Table 3.3 were designed by [Wil12] and the last five classes were developed in this thesis.

TMSE and TMRE are responsible for sending and receiving packets to or from kernel space. Both classes can be seen in Figure 3.4. TMSE checks all packets in the sending queue for the time stamp, when they have to be delivered. When this time stamp is reached or exceeded, the

²The terms of *self-induced congestion* and *congestion between chirps (inter-chirp congestion)* are used interchangeably in this thesis.

Shortcut	Meaning	Thread
MAIN	Main Program	✗
MEME	Measurement Methods	✗
TMSE	Measurement Sender	✓
TSDD	Measurement Sender Logging	✓
TMRE	Measurement Receiver	✓
TRDD	Measurement Receiver Logging	✓
TSSE	Safeguard Sender	✓
TSRE	Safeguard Receiver	✓
GPSH	Global Position System Helper	✗
NTPH	Network Time Protocol Helper	✓
RECA	Result Calculator	✗
SICD	Self Induced Congestion Detector	✗
CDTI	Countdown Timer	✓

Table 3.3: Shortcuts used in log files

packet obtains the current time stamp and is given to the kernel space by a system call. After this a copy of this packet is enqueued into the logging queue. The task of filling the sending queue belongs to the current measurement. Therefore, this thread must have enough CPU time. When TMRE received packets, the received time stamp is added and the packet is enqueued into the reception queue.

Because TMSE and TMRE are only responsible for sending and receiving, TRDD and TSDD are accountable for all data, which were pushed into the queues. TSDD approaches all sent packets and writes information into a log file. TRDD does the same for received packets, but it also operates as an interface for the main thread. Hence, not only the data is logged, also a method for calculating recent data rate is called.

If the safeguard channel is enabled, there are two additional threads. TSRE and TSSE are interfaces for the safeguard channel and should serve for a fast communication in the case of congestion or timeouts. When a packet is received, no time stamp will be added and the packet will not be logged. It is very important, that the packet is delivered to the main thread as fast as possible. The developer has the choice, whether he wants to log the data.

The part with dashed lines of Figure 3.4 is the measurement method with all kind of new classes and threads, whereby the “helper classes” are optional. Only a few interfaces have to be implemented, when a new measurement method is designed. These interfaces are listed in Table B.1 in Appendix B.1.

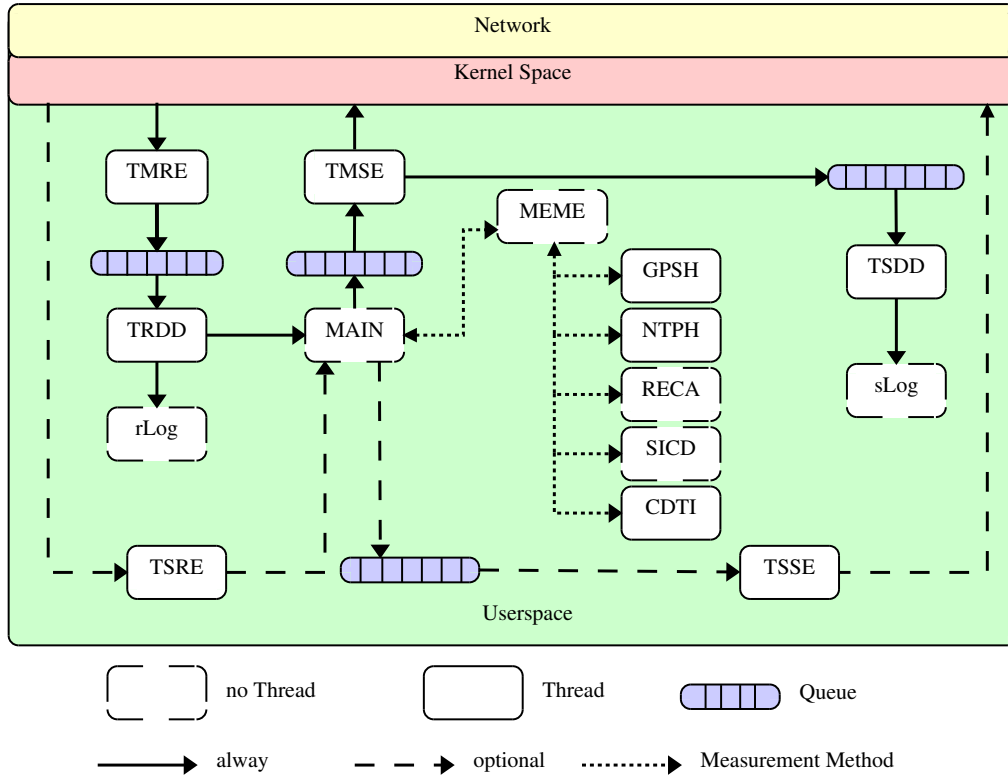


Figure 3.4: Threads, queues and helper classes of the RMF (based on [Lan13, page 10])

3.3 Synchronization and Positioning Systems

Synchronous clocks are highly important for estimating latency in data rate measurement methods. However, computer clock do not have a steady clock pulses, because environmental influences could have affects on the clock pulse (see [Goe10] and [Lan13]). Hereafter the *Network Time Protocol* (NTP) and *Global Positioning System* (GPS) by *Navigational Satellite Timing and Ranging* (NAVSTAR) are describes plus the use of both. In this thesis the term GPS will be used as acronym for NAVSTAR-GPS. At least further satellite systems will be mentioned.

3.3.1 NTP – Network Time Protocol

As described in [Goe10, page 10 f.] it is difficult to adjust one single clock, so it is not surprising, that synchronizing multiple clocks in networks is more difficult. Therefore, NTP was developed (see [Mil85], [Mil88], [Mil89] and [Mil92]). NTP is required for estimating latencies while measuring. Here the “Network Time Synchronization Research Project” offers an implementation of

commendation, which can be used to synchronize against a time server in the internet. Also, it is possible to use a local time server or GPS-mouse. While using a local or global server for time synchronization, the distance of the server is denoted as *stratum*. A stratum 1 server is directly connected to a source of commendation, which can be an atomic clock or GPS-mouse. While using such a stratum 1 device for synchronization, the terminal is a stratum 2 device. Therefore, it is obvious that the mobile measurement terminal should be a stratum 1 device, because time synchronization in cellular networks is not suitable for this thesis (see [Goe10, page 11], [KM07]) How NTP is used, can be seen in Section 4.3.

3.3.2 GPS – Global Positioning System

GPS is a satellite-system for navigation utilizing 24 satellites controlled by the USA. Different information can be received from every satellite, for example calculating the position, speed and time of the receiving device (see [NCOfSBPNT14]). In this thesis GPS is used for time synchronization and position estimation of the mobile node.

Every GPS satellite continuously sends messages, which contain the information mentioned above. Ephemerids are sent every 30 seconds and they contain time signals, parameters for calculations of the orbit and other values. In the best case, information is received from at least three satellites, because then the position of the node can be calculated with triangulation. With more than three nodes, the current time can be synchronized and in the average case 12 satellites are visible (see [Lin13]).

A GPS-receiver processes, inter alia, ephemerids and estimates a signal, which tags the start of a second. This signal is called pulse-per-second signal (PPS) (see [Mil14b]). When this signal is received by the node, an interrupt is triggered and the recently arised offset in time can be corrected. Thereby this information can be used, a GPS-mouse and a daemon is needed. The GPS-daemon *gpsd* is able to interpret the signals of different GPS-mouses over shared memory and delivers them to a TCP-port (see [RKM⁺14]). It is possible to forward PPS from *gpsd* to NTP, so that the terminal can be a stratum 1 server. Additionally, all mentioned information can be easily fetched with C++ or the GPS-traces can be logged with *gpspipe* (see [Mil14a]). For *gpsd* it is important to have a 3D-fix, because with a 2D-fix no time synchronization is available and a 1D-fix provides non valid data. A 3D-fix contains data about latitude, longitude and altitude, whereby a 2D-fix does not contain data about the altitude.

3.3.3 Galileo, GLONASS and others

GPS is not the only navigation system which can be used. Russia and Europe launched satellites for navigation too, to be independent of USA'S NAVSTAR-GPS. Even China³, Japan⁴ and India⁵ schedule their own local system.

Europe Europe launched *Galileo* as an independent, civil and global navigation system. It is controlled by the European Global Navigation Satellite System (European GNSS). The functionality is similar to NAVSTAR GPS and since January 2011 the first four satellites are in space. The whole setup is set for 2018 (see [ESA14a] and [fVudI14]), hence it is not in operational mode.

Russia The *Global Navigation Satellite System* (GLONASS) is a space-based satellite navigation system operated by the Russian Aerospace Defence Forces. The independent development was started during the Cold War for military reason. Since October 2011 the system has reached a totally global cover and therefore is an alternative to NAVSTAR GPS (see [ESA14b] and [Dal93]).

In conclusion, not only NAVSTAR GPS meets the two criteria for synchronizing clocks and tracking traces, but GLONASS can be used, too. Gpsd can process information of all three systems (see [RKM⁺14]) and is able to log the origin of the fetched datum with a suitable GPS-mouse (like the eTrex 30 Topo or Oregon devices by Garmin).

3.4 Hardware for Mobile Measurements

All measurements are done with a mobile client and a stationary server. In [Lan13, page 27] the single board computer Alix6F2 was applied as mobile client and was developed and tested successfully. [Lan13] also contains details about the used operating system called Voyage (see [DC14]), which is a derivative of Debian [itPI14]. Due to the small size of the Voyage-image, no developer libraries are included. Therefore, cross compiling was done in a virtual machine (see [Lan13, page 35]). Unfortunately, Debian does not longer support the GNU C library, which is

³*Beidou* is available for great parts of Asia and the pacific (see [CSNO14]).

⁴*Quasi-Zenith Satellite System* (QZSS) will not be in operation before 2018 (see [QSS14]).

⁵*Indian Regional Navigation Satellite System* (IRNSS) and even for local use only (see [Ind14]).

needed by the GPS-library *libgps*⁶. To deal with the problem, experimental packages were used, but these did not satisfy the dependencies of the libraries. Additionally, the RMF was improved by using circular queues. Due to this improvement a memory leak was detected, whereby *valgrind* (see [ABBH⁺14]) with *massif* did not run on Voyage. Therefore the RMF was analyzed with *Google Performance Tools* [Inc14] and *GProf* [FS14] (the output of *gprof* can be analyzed with *gprof2dot* [Fon14]). According to these problems with Voyage, *Ubuntu 14.04* 32 bit is to be used for the Alix-boards. Ubuntu offers the advantages of up-to-date libraries and has no complications with the GNU C library. In addition *libgps* is available in a recent version, but note that for compiling the RMF the packet *libgps-dev:i386* has to be installed. In addition to this, further configurations were done, whereby the previous configuration can be seen in [Lan13, page 67–72], so PPS can be used for time synchronization and the board can handle two mobile network connections at once, which is needed for the measurement- and safeguard-channel.

3.5 Traffic Generators

A good traffic generator is essential for experiments under laboratory conditions. The generator is responsible for cross-traffic and therefore simulating fallbacks and handovers (see Section 3.1). Hence, a stressed link can be simulated and therefore congestion can be simulated. This is necessary, when the adaptive properties of the novel algorithm as well as the detector for self-induced congestion will be tested and optimized. The architectural features and the abstraction layer of traffic generators can be divided in three classes (see [BDP10]):

- **Application-level traffic generators:** These generators are emulating the behavior of different network applications. For example Brute [BGPS05] and KUTE [ZKA05].
- **Flow-level traffic generators:** This type of generators will produce a replication of realistic traffic only at the flow level. For example Iperf [Kul14] and Netperf [J⁺96].
- **Packet-level traffic generators:** This kind of generator is based on packet size and packets inter departure time, the same as “gaps” in subsection 5.5.2. Most of the traffic generators are working here, for example MGEN [AG], RUDE and CRUDE [LSP02], D-ITG [APV04], UDP-gen [Zan] and MXTraF [KGL].

For controlled testing and evaluating of the novel measurement algorithm, a generator with a *constant bit rate* (CBR) for UDP traffic is needed. CBR means that a constant, specific bit rate will be generated over a given time. In contrast to CBR there are generators, where the traffic is

⁶Debian adopts Eglibc instead of Glibc now (see [Yoc14]).

for example Poisson distributed. [BDPGP12] compares different CBR-generators. In this thesis a closer look at the most cited generators on Google Scholar⁷ will be given (see [BDP10, table 1]) as well as *Brute*, because it operated safely and reliably in a previous master thesis at the institute (see [Lan13]). Additionally, the Unix standard program *Iperf* will be used.

MGEN The *Multi-Generator* (MGEN) in version 5.0 supports TCP and UDP traffic with constant bit rate for IPv4 and IPv6 networking (see [Nav14]). More features are available but are not relevant here.

D-ITG The *Distributed Internet Traffic Generator* (D-ITG) is a set of tools for reproducing and analyzing internet traffic. It generates CBR with UDP and TCP as well as VoIP or telnet traffic (see [APV04]). Even normal- or Poisson distribution can be simulated. Therefore, traffic according to the real world can be created and used for testing the novel algorithm. Also game traffic as Quake 3 and Counterstrike was analyzed and can be generated (see [LBA04]). Unfortunately duplex mode is not available in D-ITG version 2.8.1 and so it only is for one-way traffic generation.

RUDE and CRUDE RUDE stands for *Real-time UDP Data Emitter* and only generates UDP network traffic. CRUDE is the *Collector for RUDE*, thus CRUDE can receive and log RUDEs generated traffic. Both programs are available in version 0.7.

Iperf Iperf – in version 2.0.5 – is a standard tool, which is included in many Linux distributions like Ubuntu and Debian. It estimates the current available data rate on a given link through traffic generation. In addition the size of generated traffic can be set as parameter, too. Additionally, Iperf has got all options included for creating a UDP- or TCP-traffic-flow, even for a given time and duplex mode.

Brute Brute was used by [Lan13] and is a high performance and extensible traffic generator. In comparison to GEN and Rude it achieves the highest maximal throughput achievable with CBR. Unfortunately, there is no documentation of Brute on [Bon14] and only the paper [BGPS05] exists. Additionally, there is a bug in automake-1.14. Therefore, automake 1.9 was used and line 107 in `src/Makefile` was changed to `MKDIR_P = mkdir`.

⁷<http://scholar.google.de/>

Chapter 4

Framework Improvements

This chapter explains improvements for the framework, which were made while the new algorithm was developed. First Section 4.1 briefly describes a wrapper for posix threads. Section 4.2 and 4.3 deals with wrappers for GPS-signals and NTP synchronizing. Afterwards Sections 4.4, 4.5 and 4.6 briefly explains the classes for calculation of the data rate, a simple countdown time and the detector for self-induced congestion. Additionally, the measurement algorithm displays status reports while measuring. These reports contains information about the enqueued chirp as well as current values for sleeping due congestion.

4.1 Posix Thread Wrapper

The main part of the Rate Measurement Framework was written shortly after C++11 was introduced but still not supported by most compilers. Therefore, the RMF is based on C++ *standard technical report 1* which does not provide an easy way to handle posix threads. To ease porting the RMF, the boost library has not used. Therefore a wrapper class was implemented during this thesis. The ideas are based on [Bou14].

4.2 GPS Helper Class

The GPS helper class extends the posix thread wrapper, fetches and logs GPS data periodically with the help of the GPS-daemon *gpsd*⁸ or automatically with the tool *gpspipe* (see [Mil14a]).

⁸Whereby the library `libgps20` or higher is necessary

In this thesis both possibilities will be used. All fundamentals about GPS were briefly described in section 3.3.2. For the helper class `libgps20` or higher is needed. If the user sets parameter for using `gpsd` respectively `gpspipe`, the GPS helper class will periodically check, whether both programs are running. Thereto the status of `gpsd` can be checked by a system call. By contrast `gpspipe` can be checked with the help of its *program identity number* (PID) . If the status check of `gpsd` or `gpspipe` fail or at least one of them is not running, the class tries to restart it. Important is, that `gpspipe` has to be killed manually, when the RMF fails, because `gpspipe` is a system process.

The `gpsd` exports data in three ways: First way is via a sockets interface, secondly via DBUS broadcasts and the last way is via a shared-memory interface. Because the RMF is written in C++, shared-memory can be used as entry point. Via shared memory, two kinds of reports can be received. First one is a *time-position-velocity* (TPV) report and the second one is called *SKY* report. The TPV report contains all estimated information, like time, GPS coordinates and some error values. A sky report contains the sky view of the GPS satellite positions and therefore it is very important for GPS. The main interests for the measurement method are in time, longitude and latitude. A sample can be seen in Listing D.1. For a closer look, Listing D.2 contains data, which were saved by `gpspipe` in json-format.

GPS Dates As just mentioned, data is received via shared-memory. For an easy-handling, the data is parsed and the values are saved as `gps-date-object`. Therefore, `gps-date` is a class for handling and saving values from exactly one GPS-date, which can be extracted from `gpsd` data. Additionally, the measurement method receives a GPS-date via callback every time, when the GPS-helper class handles GPS-data.

4.3 NTP Helper Class

The importance of NTP was already mentioned in Section 3.3.1. The NTP helper class extends the `posix` thread wrapper for checking NTP periodically, too. Therefore, the program `ntpq` (see [Kel14b]) is used. But before the values, which are delivered by NTP, can be checked, the terminal has to use the PPS for synchronizing against this signal. [Lam14] offers an instruction for time synchronization with a Garmin GPS. In this thesis a *Garmin OEM 18x LVC* was used, circuit diagrams and more information are in [Goe10, page 17]. Finally, the NTP configurations of both terminals has to be edited. The changes can be seen in Listing C.6 and C.7. Additionally, the NTP log files should be kept by the operation system, so Listing C.8 describes the edited cron job.

Unfortunately, there was no possibility for placing the server near a window, hence the time servers of the data center of the University of Dusseldorf had to be used. Listing D.3 shows the data, which are get with a system call of `ntpq -p`. The system call was done a a laptop, which sued both of time servers of the data center. SHM describes the shared memory section, where the PPS is fetched from the operating system. Current selected server is marked with *, while additional acceptable servers are marked by an +. Both server are included in the weighted average computation to set the local clock of the terminal (see [Kel14a]). - denotes a source, which is discarded by the algorithm. Interesting columns are “reach” and “jitter”. First one is an octal shift register, which can be 377 in the maximum, so the last eight requests were successful ($377_8 = 1111111_2$). The jitter column denotes the difference between two samples in milliseconds, therefore values near zero are perfect. The time servers only differ minimal from the PPS resource, therefore they are appropriate for time synchronization. Keep in mind, that the client has to use PPS, because time synchronization with NP in cellular networks will fail (see [KM07]). Every 15 seconds the NTP helper class checks for a small value in the jitter column (smaller than 100 ms) and large value in the reach column (greater than 0). So a boolean value will be callbacked to the measurement method. If the jitter-value is too big or reach-value too small, only a error-message will be displayed, because the helper class will manage the restart of NTP. Unfortunately, the measurement results will be corrupt, when the time is not correct. However this can be seen in the log files of each measurement or in the peerstats of NTP.

Figure 4.1 clarifies the differences in offsets between the local clock and PPS as well as both time servers (RZ1 and RZ2) of the data center in August 2014. For the datasets of the entire month, median, minimum and maximum values are shown. Every data was fetched by an laptop with the Garmin GPS. Figure 4.1a shows the entire August, whereby there is a spikes at the 11th and 12th of August. That is why Figure 4.1b shows the same data, only zoomed in. For a clear presentation, the median is hide, but the values are approximately near -0.2 ms. As can be seen, the offset of the time servers is smaller than half a millisecond. Details about the spike of the 11th August are shown in Figure 4.1c and 4.1d, whereby the 12th of August is shown in Figure 4.1e and 4.1f. As can be seen, first spike only is a inaccuracy and can be neglected. The second spike is a inaccuracy of the time values themselves and even the employees of the data center had no excuses. That is why the NTP peerstats for September were logged and can be seen in Figure 4.2. In September there is an imprecision for the third day, which can be seen in Figure 4.2c and 4.2d. For a clear presentation, the median is hide, but the values are approximately near -0.2 ms, too. The PPS drifts, because the notebook with the GPS mouse was restarted. The drift for RZ1 and RZ2 is based on an error of the address resolution protocol (ARP) of the time servers. Summarized the time servers are suitable for synchronizing.

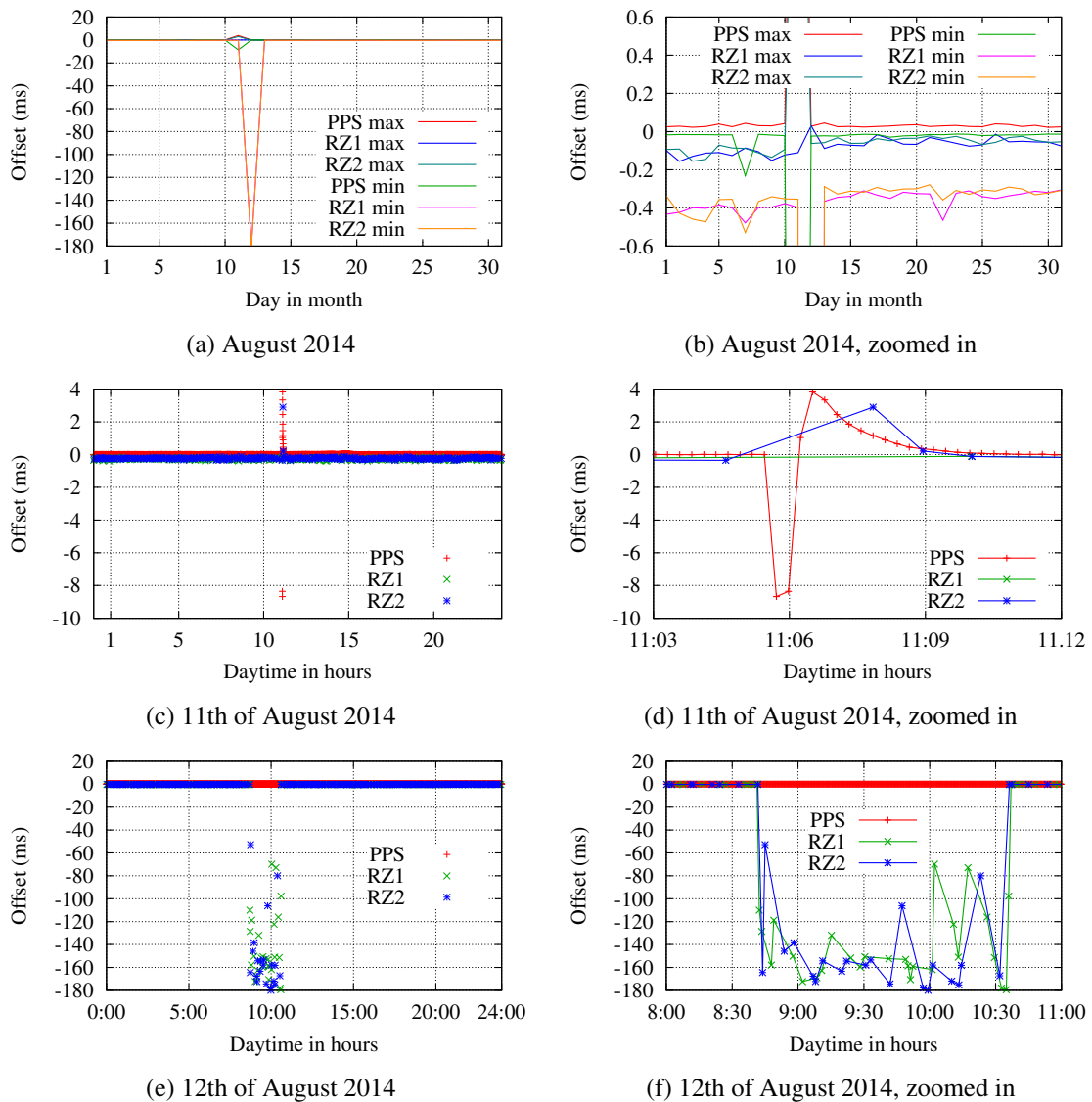


Figure 4.1: NTP offset from PPS and time servers of the university for August 2014

4.4 Result Calculator

The class of the result calculator was part of the measurement method itself, but this was not convenient. There was no option for estimating results, when the measurement was canceled through errors. Therefore, now a result calculation can be run without a previous measurement. Originally the estimation has smoothed the results with three values, now there could be set a parameter, how many values should be taken for smoothing. Lastly additional plot files will be generated, like debugging data for the safeguard channel or for checking loss.

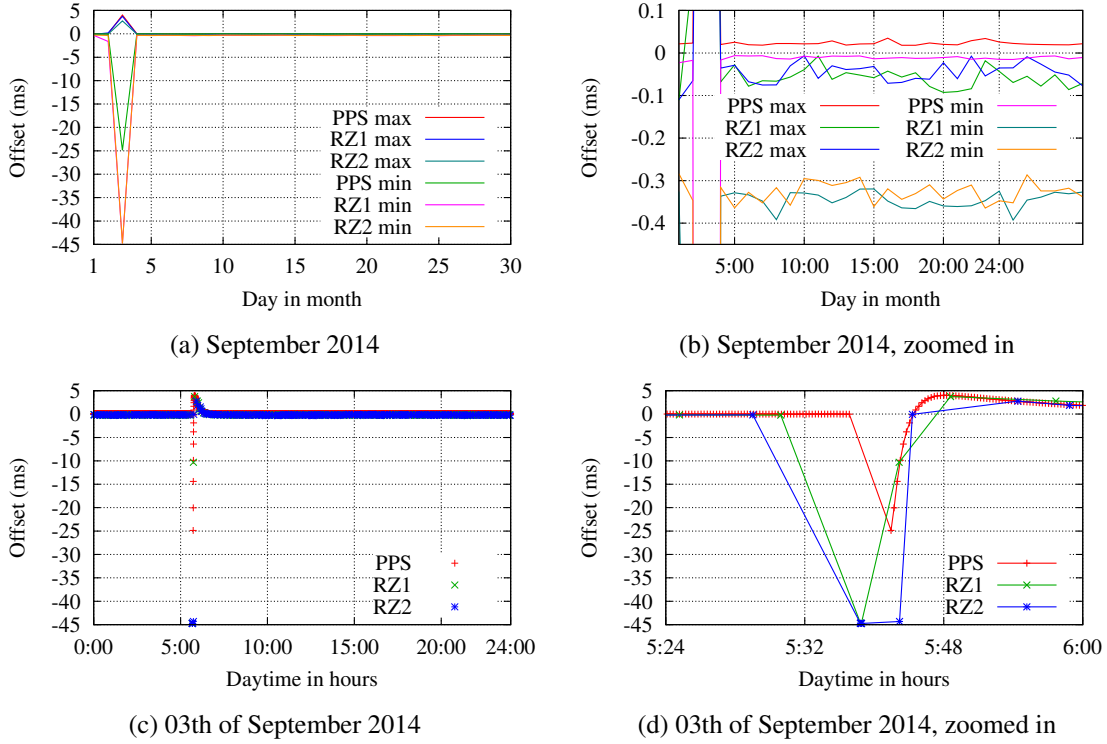


Figure 4.2: NTP offset from PPS and time servers of the university for September 2014

4.5 Countdown-Timer

Even this class extends the posix thread wrapper. For example the countdown-timer could be set for sending a message via safeguard-channel at a specific time or a specific timeout. Additionally, reasons for the timer could be set, so that on timeout the reason will be transmit via callback.

4.6 SICD – Self-Induced-Congestion Detector

SICD is the main part of the new algorithm. This class contains two mechanism for detecting congestion. First mechanism will check for congestion at the beginning of a chirp and the second mechanism investigates congestion within current chirp. If congestion is detected, a sleep time will be calculated. This time denotes a pause for a terminal. If the congested terminal will pause for this time, the phase of congestion will be gone and the terminal can start sending again. All details will be explained in section 5.5.

Chapter 5

Novel Measurement Algorithm

This chapter introduces the novel measurement algorithm which is capable of measuring available data rates, delays and drop rates separately for each direction of a network path. The development of the new algorithm was focused on 2G and 3G cellular networks (see Section 3.1), because it is commonly used in C2C. Therefore, the used modem had a maximal available data rate of 7200 Kbit/s. The introduction is divided in five parts and they are divided as follows:

Section 5.1: Assumptions for the measurement.

Section 5.2: Starting a measurement.

Section 5.3: Sending packets via measurement- and safeguard-channel.

Section 5.4: Receiving measurement- and safeguard-packets as well as calculate data rate.

Section 5.5: Check for self-induced congestion.

5.1 Assumptions

Section 3.3 explained the need for GPS and NTP. Unfortunately, both have to be started before the measurement will be done, because they need time to get signals. Whereby GPS is faster than NTP, because the GPS mouse only has to get ephemerids from the satellites (see Section 3.3.2). NTP requires a integration phase up to 30 minutes (see [Goe10, page 43] and [Goe10, Figure 4.2]). Therefore, the NTP service and GPS daemon have to be checked, before a measurement can be started. Furthermore measurements will be done every second, within an 1 Hertz interval and the channel load is set to 50 %. This means that packets will be send with infinite bandwidth, so that the other terminal is able to receive the packets within 500 ms. With this utilization, drops of the data rate up to 50 % do not have a negative influence on the measurements. Addition,

the first packet is very important for the measurement, because it has the only correct sending time stamp under the assumption of an empty network queue (keep in mind that packets are send back-to-back). Therefore, the time stamp of this packet is highly important for calculations and due to the risk of loss, the time stamp will be piggybacked. Further the first packet is smaller than the others, because it is insignificant for the estimation of the data rate. Last but not least every data rate is calculated with a base of 1000.

5.2 Starting a Measurement

When all assumptions are accomplished, a measurement can be started. The general process was already described in Figure 3.2, but the content of the second synchronization message of the client has not been mentioned yet. Second synchronization message has the prefix `MEASURE_ALGO_SYNC`, followed by values, divided by the pipe symbol. More about the default values in Section 6.1.3.

The measurement starts with a “slow-start” phase of two seconds respectively two chirps. The first chirp is based on a starting data rate of 16 KByte/s (see Section 6.1.3). Though the backchannel is empty in the beginning and the first piggybacked values are received after the first chirp. After this the estimated data rate can be adapted, so that the second chirp will be send with the estimated data rate for the last chirp.

	Dynamic Part	Fixed Part
Bytes	4 Bytes	4 Bytes
0-3	based on id	packet id
4-7	data rate (integer part)	(chirp length \ll 16) + packet size
8-11	data rate (fraction part)	send time of first packet in chirp (integer part)
12-15	loss in percent	send time of first packet in chirp (fraction part)
16-19	time to sleep (integer part)	
20-23	time to sleep (fraction part)	
24-27	based on id of time to sleep	

Table 5.1: Payload data of the measurement packet

Table 5.1 illustrates the dynamic and fixed payload data of the measurement packets. The backchannel is realized with the dynamic payload, whereby non-changing values will be saved in the fixed part. Data rate and time stamps are splitted in an integer and a fraction part due to high precision. Chirp length and packet size can be send as one integer with the formula given in Table 5.1. The packet size has to be piggybacked, because the first packet has a different size

than the rest of the chirp and the “normal” packet size is needed for calculations in SICD. *Time to sleep* is the time the receiver has to be quiet for reducing inter-chirp congestion. This value will also be send in the safeguard channel, but the safeguard channel is optional and it is not assured that the safeguard packets are received faster.

5.3 Sending packets via measurement- and safeguard-channel

In this thesis the following terms are be used: *Chirp length* denotes the count of packets per chirp, *chirp size* equals the added size of all packets of one chirp and *packet size* denotes the size of one packet, thus it differs between chirps and the first packet always is an exception. After calculating chirp length and packet size, the chirp can be generated. Packets in one chirp but the first only differ in their ids and time stamps except for the payload data. Without loss of generality a chirp will be send once per second. Therefore the sending methods sleeps for 10 milliseconds until 1 second has passed. Due to this small sleeping time, the method can find out, whether congestion occurs. In this case, the method will wait longer than 1 second. This is explained in Section 5.5.1. Afterwards the packets are generated, they are given to the sending network queue of the measurement channel, the rest is done by the framework. While packets are send back-to-back, congestion occurs. This *packet congestion* is used for estimating data rate, latency and drop rate, but *inter-chirp congestion* should be avoided. The sender estimates the chirp length and packet size in a way so that the other terminal should receive the current chirp within 500 milliseconds. Due to a buffer of 50 % of the current available bandwidth, drops of the data rate of maximal 50 % could be intercepted. When the transmission needs more then one second, the time stamp of the first packet of next chirp will be malicious and unsuitable for estimating the current backbone delay. After the detection the receiver has to notify the sender. Therefore, packets via the safeguard channel will be send.

Table 5.2 illustrates the dynamic payload of the safeguard packets, whereas the send time stamp needs to be piggybacked in the safeguard packet, because safeguard packets do not provide any send time stamp. The send reason stores information, why this packet was send. Reasons are congestion, loss or timeouts. A timeout denotes the gap in time, when no packet has received since half a second plus a buffer of 100 milliseconds. This case servers as hedge against fast changing network properties. In the integer of the send reason just one bit will be set, because the receiver of this packet can check the reasons with a bit mask (see Table 5.3). Due to the limited scope of this thesis, influences of loss on the measurement could not be considered, but the implementation offers many templates for this problem. Additionally the recent estimated data

Bytes	4 Bytes
0-3	based on id
4-7	(chirp length \ll 16) + packet size
8-11	data rate (integer part)
12-15	data rate (fraction part)
16-19	send time (sec part)
20-23	send time (nsec part)
24-27	(loss \ll 16) + send reason
28-31	time to sleep (integer part)
32-35	time to sleep (fraction part)

Table 5.2: Dynamic payload data of the safeguard packets

Bitmask	Meaning
0000 0001	congestion
0000 0010	loss
0000 0100	timeout

Example: 0000 0011
High loss and congestion

Table 5.3: Bitmask for the send reason in the payload of safeguard packets

rate will be piggybacked. Thus the sender will receive its last data rate for sending, if congestion occurs and the backchannel is blocked.

5.4 Receiving Packets

Sent Packets should be received in the best case, whereat there could be loss on the link due to various circumstances. Figure 5.1 describes the different time stamps within a chirp on sender and receiver side, whereat $j.i$ denotes the i th packet in chirp j . The sender transmits a chirp with four packets, so that the chirp should be received within 500 milliseconds. During transmission packets could get lost, so just the last three packets are received. Additionally, the supposed receive time of 500 milliseconds cannot be hold due to changes in the network characteristics. Therefore, there is an additional delay of $x \in \mathbb{R}$.

After the successful reception of at least two packets, data rate can be calculated for every chirp:

$$\text{data rate} = \frac{\text{count of received packets} \cdot \text{packet size}}{\text{delay of transmission of the chirp}}.$$

If the first send packet of chirp was received, one packet has to be subtracted of the first multiplicand of the numerator. This has to be done, because the first packet is for getting the send time stamp only. The calculated data rate will be piggybacked via backchannel so the sender gets to know the available data rate while the last chirp was sent.

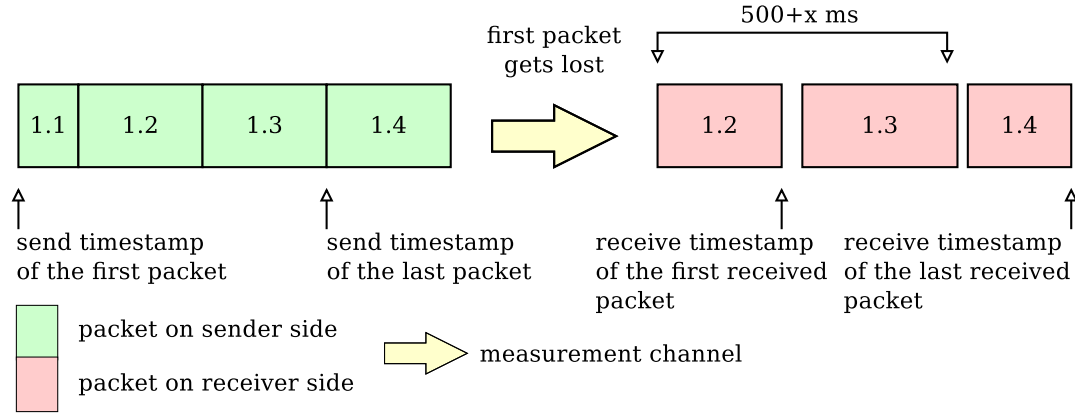


Figure 5.1: Presenting time stamps within a chirp

5.5 SIC – Inter-Chirp Self-Induced Congestion

As mentioned in Section 5.2 *self-induced congestion* respectively *inter-chirp congestion* needs to be avoided, because this kind of congestion has a negative effect on three properties of the measurement. First, the delay measurement will be destroyed, because the sending time stamps are incorrect. Second, a measurement frequency of one hertz can not be maintained, because the appeared congestion has to be reduced. Third, the piggybacked information will be received delayed. Unfortunately, prevention of SIC is not always possible due to influences like fallbacks and handovers (see Section 3.1). Hence, SIC must be detected as fast as possible. Therefore, two ways for detecting SIC were developed. Section 5.5.1 is about detecting SIC within current chirp and Section 5.5.2 describes the tracing of SIC after every first packet of a chirp. In the following, the most important notation are listed in Table 5.4, whereat the receive time stamp will be referred to tx instead of rx . It should be mentioned that the first received packet must not be the first send packet ($f \neq 1$). For the last packet it hold vice versa.

5.5.1 Self-Induced Congestion within current Chirp

Figure 5.2 describes the logic for detecting SIC within current chirp. It is presumed that $k \geq 2$ packets are received, because at least two packets are needed for calculations. Generally the duration of current chirp will be calculated, so that the time between the chirps – the inter-chirp gap time – can be checked. Hence, $t_i^{erx}(n_i)$ has to be calculated. Therefore d^{air} of all expected packets and the d_{bb} from the last non-congested packet are added to the time stamp, of the last received packet. Thereby the remaining time of the current chirp can be checked when the dif-

Acronyms	Declaration
t^{tx}	transmit time stamp
t^{etx}	estimated transmit time stamp
$t_i^{tx}(k)$	transmit time stamp of packet k of chirp i
sp_i	packet size of all packets but the first of chirp i
sp_1	packet size of the first packets
n_i	count of packets in chirp i
A_i	calculated data rate of chirp i
d_{bb}	current backbone delay
$d_i^{air}(k)$	air delay of k th packet in chirp i
f, j, k, l	are abbreviations for the first, any, the current and the last packet in a chirp, where $1 \leq j, k \leq n_i$ holds for chirp i

Table 5.4: Acronyms for time stamps and packet properties

ference between $t_i^{rx}(n_i) - t_i^{rx}(1)$ will be considered (keep in mind that $t_i^{tx}(1)$ is piggybacked and $t_i^{rx}(1)$ can be calculated). If this difference is greater than the measurement interval subtracted by d^{air} of all previous congested packets, than congestion occurs or occurred and *time to sleep* will be calculated. Otherwise all timers, flags and other variables can be reseted. The value of “time to sleep” denotes the relative time for the sender, which he has to be quiet for reducing inter-chirp congestion. Amount of packets in the current chirp, which were received in a phase of congestion, is denoted as c .

5.5.2 Self-Induced Congestion between Chirps

Figure 5.3 describes the logic of detecting SIC between chirps. For the flow chart it is presumed that the last chirp i is not congested, but chirp $i + 1$ could be congested. First, a trend in the development of the download data rate of chirp i has to be assumed. Therefore, a line of best fit based on the gaps between consecutive packets in the last chirp is calculated. When the gaps are getting bigger, the data rate is decreasing, otherwise the data rate increases. If the slope is equals zero, a positive slope is assumed and a second check will be done due to verification. This is done, because the cumulative slopes of every cut could be zero, but the last slope (or last slopes) is positive.

First, left branch of Figure 5.3 is described. In this case, the data rate is rising and the time between $t_{i+1}^{rx}(1)$ and $t_i^{rx}(n_i)$ is calculated. If this gap is lower or equal accumulated d^{air} of all outstanding packets of the last chirp, the first packet of the current chirp and the last packet of last chirp were received back-to-back and therefore congestion will occur. Then the time to sleep has

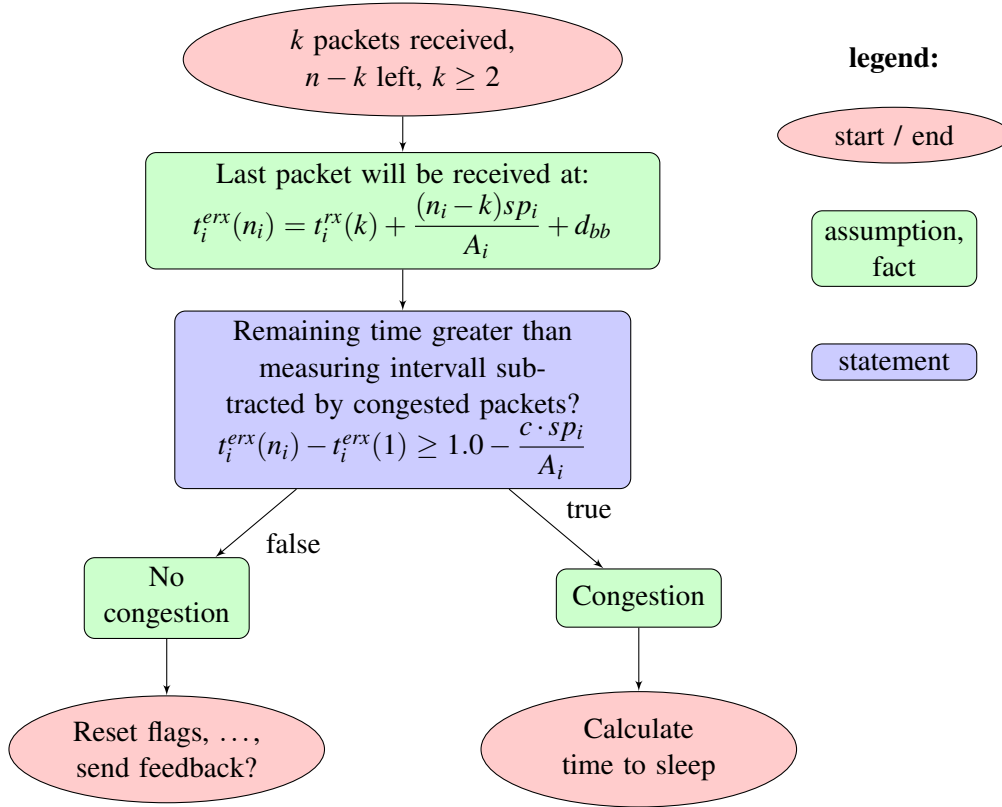


Figure 5.2: Detecting self-induced congestion within current chirp

to be calculated. If the gap is greater than the accumulated air delays, congestion will not occur and all timers, flags and other variables are reset.

The right branch of Figure 5.3 represents the case of a falling data rate. Then a second line of best fit is calculated, based on the last $\sqrt{n_i}$ packets of the current chirp. This is done because the development of all gaps could be like a “v” (or any other shape with a increasing gap size at the end). In this case the slope is equals zero, but the gaps are rising at the end and the data rate decreases. Furthermore two metrics from [RRBLC03] will be used. First one is the *pairwise comparison test* (PCT) and second one the *pairwise difference test* (PDT) (see Section 5.5.4). PCT and PDT are indicators for the trend of delays of the packets. Summarized, if the second gradient is positive or the two metrics got an increasing trend, then the data rate for the last $\sqrt{n_i}$ packets will be estimated and used for future calculations. Otherwise, the last check will be done directly. A proof of concept for the value of $\sqrt{n_i}$ can be seen in measurements in Chapter 6, where no ambiguous behaviour occurred.

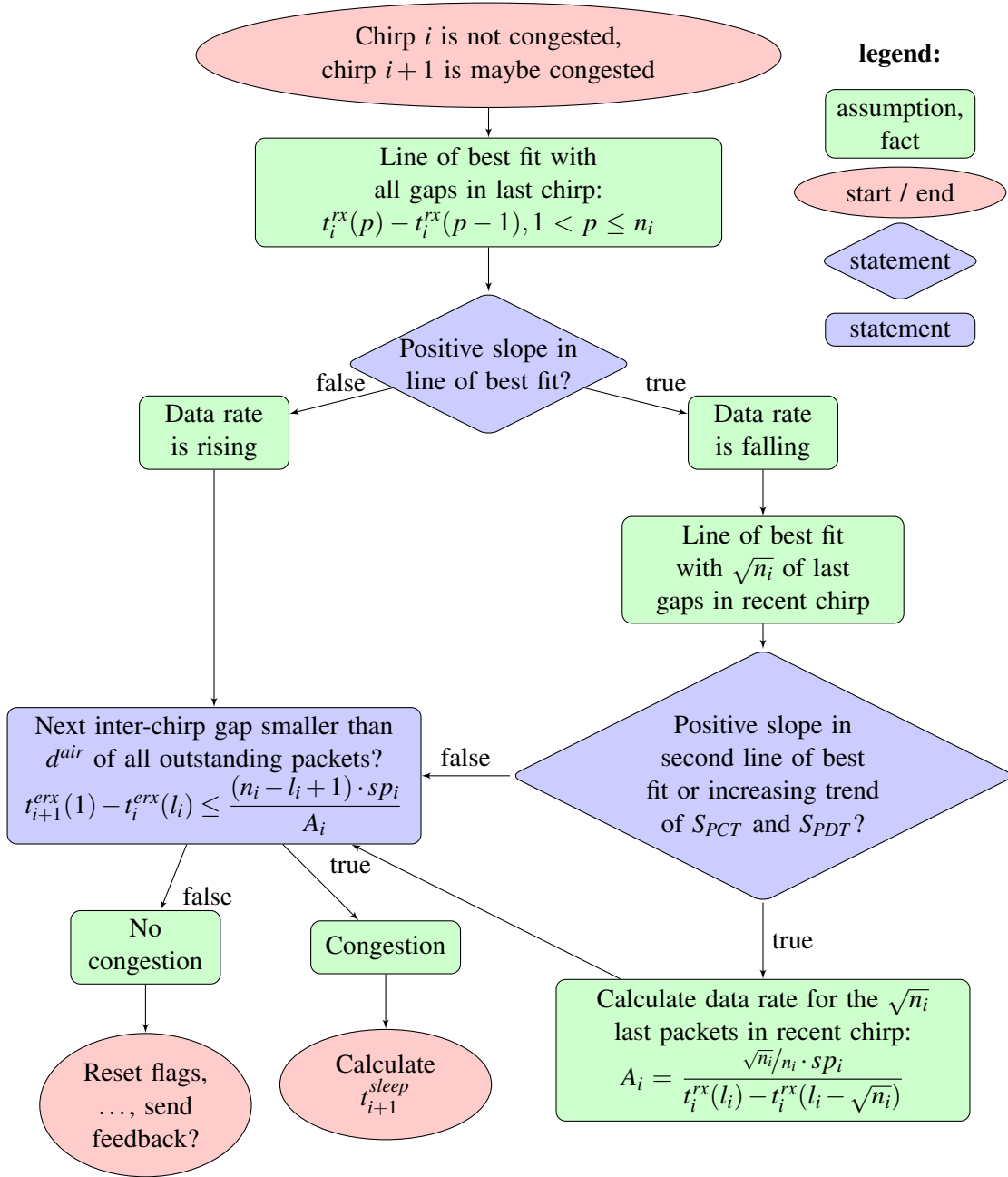


Figure 5.3: Detecting self-induced congestion between chirps

5.5.3 Linear Regression

As described in Figure 5.3 the line of best fit has to be calculated for evaluating the development of the data rate. The line of best fit can be calculated with linear regression and is defined by:

$$f(x) = m \cdot x + b \quad b, m, x \in \mathbb{R} \quad (5.1)$$

where m represents the slope of the line and b is the y-intercept.

Following equations used for simplifying the Equation 5.1:

$$\sum_{i=1}^n i = \frac{1}{2}n(n+1), \quad \text{gaussian sum} \quad (5.2)$$

$$\sum_{i=1}^n i^2 = \frac{1}{6}n(n-1)(2n-1), \quad \text{square pyramidal number.} \quad (5.3)$$

With Equations 5.2 and 5.3, Formula 5.1 can be calculated. The gradient can be defined as follows (see [Ler12]):

$$m = \frac{\sum_{j=1}^{n_i} X_j \cdot Y_j - \frac{1}{n_i} \cdot \sum_{j=1}^{n_i} X_j \cdot \sum_{j=1}^{n_i} Y_j}{\sum_{j=1}^{n_i} X_j^2 - \frac{1}{n_i} \cdot \left(\sum_{j=1}^{n_i} X_j \right)^2}$$

Now $Y_j = t_i^{rx}(j+1) - t_i^{rx}(j)$ is the packet dispersion in the current chirp, the so called *gap*. $X_j = j$ denotes the gap number in $[1, n_i - 1]$, where gap j is the gap between packet j and $j+1$. Therefore there are $n_i - 1$ gaps between n_i packets. There it is:

$$\begin{aligned} m &= \frac{\sum_{j=1}^{n_i-1} j \cdot Y_j - \frac{1}{n_i-1} \cdot \sum_{j=1}^{n_i-1} j \cdot \sum_{j=1}^{n_i-1} Y_j}{\sum_{j=1}^{n_i-1} j^2 - \frac{1}{n_i-1} \cdot \left(\sum_{j=1}^{n_i-1} j \right)^2} \\ &\stackrel{5.2}{=} \frac{\sum_{j=1}^{n_i-1} j \cdot Y_j - \frac{1}{n_i-1} \cdot \frac{n_i \cdot (n_i-1)}{2} \cdot \sum_{j=1}^{n_i-1} Y_j}{\sum_{j=1}^{n_i-1} j^2 - \frac{1}{n_i-1} \cdot \frac{n_i^2 \cdot (n_i-1)^2}{4}} \\ &\stackrel{5.3}{=} \frac{\sum_{j=1}^{n_i-1} j \cdot Y_j - \frac{1}{2} n_i \cdot \sum_{j=1}^{n_i-1} Y_j}{\frac{1}{12} \cdot (n_i-1) \cdot (n_i^2 - 14 \cdot n_i + 12)} \end{aligned} \quad (5.4)$$

The axis section b in Equation 5.1 is defined by:

$$b = \frac{1}{n_i} \left(\sum_{j=1}^{n_i} Y_j - m \cdot \sum_{j=1}^{n_i} X_j \right) s \stackrel{5.2}{=} \frac{1}{n_i} \sum_{j=1}^{n_i} Y_j - \frac{m \cdot (n_i - 1) n_i}{2 n_i}. \quad (5.5)$$

Equations 5.4 and 5.5 are used in Figure 5.3 for the linear regressions.

5.5.4 PCT – Pairwise Comparison Test and PDT – Pairwise Difference Test

In [RRBLC03] the authors introduce two metrics. PCT and PDT are metrics for analyzing the *one-way delays* (OWD) of the packets in a chirp. The PCT represents a key ratio for the trend of consecutive OWDs in the current chirp and is stable with respect towards outliers (see [RRBLC03, figure 4]). The key ratio of PDT quantifies the strength of the start-to-end gap variation, an example is given in [RRBLC03, figure 4], too. Both ratios can be calculated as follows, where all gaps Y_j are partitioned in $k = \lceil \sqrt{n_i} \rceil$ groups. The median of the group p with $1 \leq p \leq k$ is \hat{Y}_p with:

$$\hat{Y}_p = \hat{Y}_j \left\lfloor \frac{k+1}{2} \right\rfloor$$

S_{PCT} and S_{PDT} denote the metrics for PCT and PDT, whereat S was used as an acronym for “stream” in [RRBLC03] and just adopted. It is $0 \leq S_{PCT} \leq 1$, $-1 \leq S_{PDT} \leq 1$ and:

$$S_{PCT} = \frac{\sum_{j=1}^{n_i-1} I(\hat{Y}_{j+1} > \hat{Y}_j)}{k-1} \text{ and } S_{PDT} = \frac{Y_{n_i} - Y_1}{\sum_{j=1}^{n_i-1} |Y_{j+1} - Y_j|}.$$

Here $I(X)$ is one if X holds and zero otherwise. The authors state that S_{PCT} is increasing in [JD02], when $S_{PCT} > 0.66$. For S_{PDT} it is $S_{PDT} > 0.55$. The development of gaps is increasing when both ratios are increasing. Both values were used for the new algorithm, as seen in Figure 5.4. The values in this figure are out of the measurement in Figure 6.13. It is important, that congestion was detected by the server after the 5 s, 23 s, 36 s, 52 s, and 109 s. Therefore, PCT and PDT provided indications for congestion in 4 out of 7 occurrences of inter-chirp congestion.

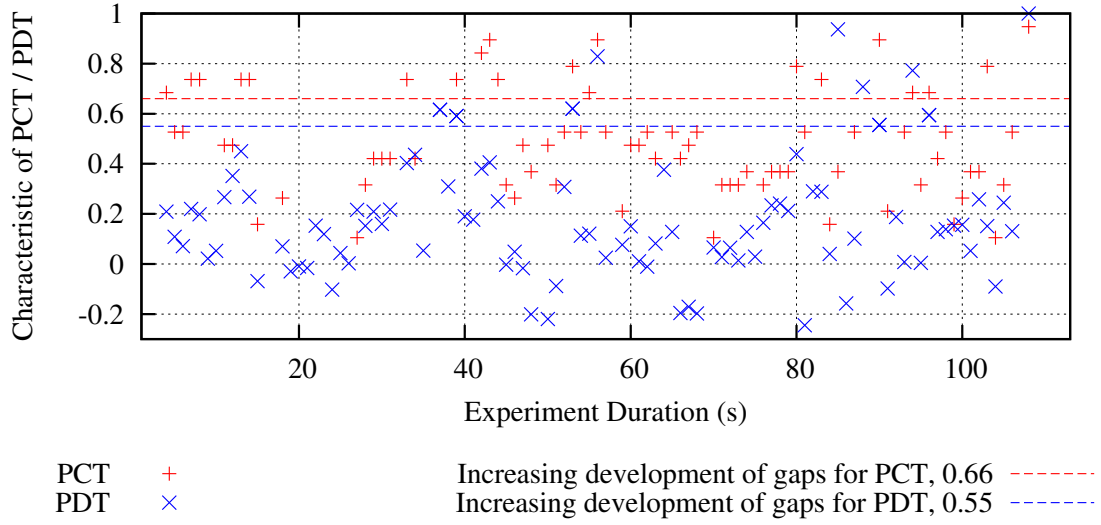


Figure 5.4: Values of PCT and PDT for measurement in Figure 6.13

5.5.5 Calculation of transmit time of the next chirp

If SIC is detected, the send time stamp of the first packet of the next chirp has to be calculated. This time stamp is defined as: $t_{i+2}^{tx}(1) = t_{i+1}^{rx}(k) + t_{i+1}^{sleep}$. The value of t_{i+1}^{sleep} is the *time to sleep*, or the time where no further packets should be send. Figure 5.5 clarifies the necessary time stamps for this subsection. Calculating $t_{i+2}^{tx}(1)$ is the same as calculating t_{i+1}^{sleep} , only the processes are slightly different, because $t_{i+2}^{tx}(1)$ is an absolute and t_{i+1}^{sleep} a relative time stamp. In this thesis

$$t_{i+1}^{sleep} = t_{i+2}^{tx}(1) - t_{i+1}^{rx}(k) \quad (5.6)$$

will be calculated.

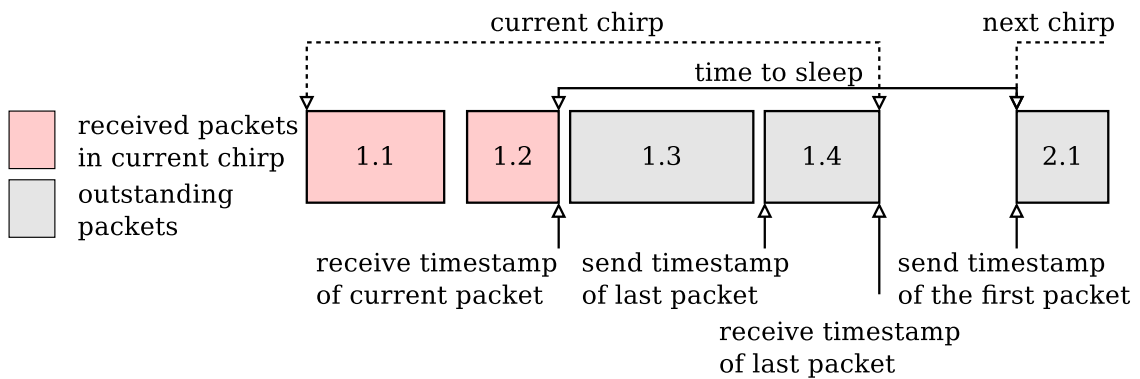


Figure 5.5: Presentation of “time to sleep”

For Formula 5.6 the value of $t_{i+1}^{rx}(k)$ is well known, because it is the receive time of the current packet. Hence, $t_{i+2}^{etx}(1)$ has to be estimated. There $t_{i+1}^{etx}(n_{i+1})$ is needed, and therefore the approximated air delay $d_{i+1}^{air}(n_i) = \frac{sp_{i+1}}{A_i}$ of all outstanding packets has to be added to $t_{i+1}^{rx}(k)$. The value of d^{air} is approximated, because the current data rate A_{i+1} is unknown, only the data rate of the previous A_i is known. Summarized it is:

$$\begin{aligned} t_{i+1}^{etx}(n_{i+1}) &= (n_{i+1} - k) \cdot d_{i+1}^{air}(k) + t_{i+1}^{rx}(k) \\ &= (n_{i+1} - k) \cdot \frac{sp_{i+1}}{A_i} + t_{i+1}^{rx}(k). \end{aligned} \quad (5.7)$$

When estimating $t_{i+2}^{etx}(1)$ the value of $t_{i+1}^{etx}(n_{i+1})$ is needed. This value can be calculated with $t_{i+1}^{etx}(n_{i+1})$. Therefore, $d_{n_{i+1}}^{air}$ is subtracted from the receive time stamp of the last packet in the current chirp. Additionally, d_{bb} is always the backbone delay of the last non-congested packet. The value of $t_{i+1}^{etx}(n_{i+1})$ is defined by:

$$\begin{aligned} t_{i+1}^{etx}(n_i) &\approx t_{i+1}^{etx}(n_{i+1}) - d_{i+1}^{air}(n_{i+1}) - d_{bb} \\ &= t_{i+1}^{etx}(n_{i+1}) - \frac{sp_{i+1}}{A_i} - d_{bb} \end{aligned} \quad (5.8)$$

The unknown backbone delay can be calculated from the last non-congested packet, which is – without loss of generality – the first received packet of the previous chirp. Generally it is

$$d_{bb} = t_i^{rx}(1) - t_i^{tx}(1)$$

but the reception of the first packet of any chirp is not guaranteed. Therefore, the first received packet can be taken, whereas the air delays of all previous packets have to be subtracted. Keep in mind that the send time stamp of the first packet can be piggybacked, so it is:

$$\begin{aligned} d_{bb} &= t_i^{rx}(f) - t_i^{tx}(1) - d_i^{air}(\text{“all previous packets”}) \\ &= t_i^{rx}(f) - t_i^{tx}(1) - d_i^{air}(1) - (f - 1) \cdot d_i^{air}(2) \\ &= t_i^{rx}(f) - t_i^{tx}(1) - \frac{sp_1}{A_i} - (f - 1) \frac{sp_i}{A_i}. \end{aligned} \quad (5.9)$$

Now the estimated send time stamp of first packet of chirp $i + 2$ can be estimated:

$$\begin{aligned}
 t_{i+2}^{etx}(1) &= t_{i+1}^{etx}(n_{i+1}) + d_{i+1}^{air}(n_{i+1}) + t_{i+1}^{gap} \\
 &\stackrel{5.8}{\approx} t_{i+1}^{erx}(n_{i+1}) - d_{i+1}^{air}(n_{i+1}) - d_{bb} + d_{i+1}^{air}(n_{i+1}) + t_{i+1}^{gap} \\
 &= t_{i+1}^{erx}(n_{i+1}) - d_{bb} + t_{i+1}^{gap}. \tag{5.10}
 \end{aligned}$$

Here t_{i+1}^{gap} denotes the gap time between chirp $i + 1$ and $i + 2$. Keep in mind that the channel utilization is 50 %, so that send packets are received for half a second and the other half nothing is send. The second half serves as inter-chirp gap time and protects the measurement from congestion due to data rate drops up to 50 %. Unfortunately, the data rate could drop by more than the half, leading to inter-chirp congestion. Figure 5.6 explains the importance of the gap time. In the desirable case, the sender can transmit the chirp, so the other terminal receives this chirp within 500 milliseconds. If the available data rate is high enough, packets will not be congested (respectively, packets will not be queued due to a too high sending data rate) and can be received in less than the measurement interval. Nevertheless congestion can occur, so that the chirp latency is higher than one second. This case is shown as congested behaviour in Figure 5.6.

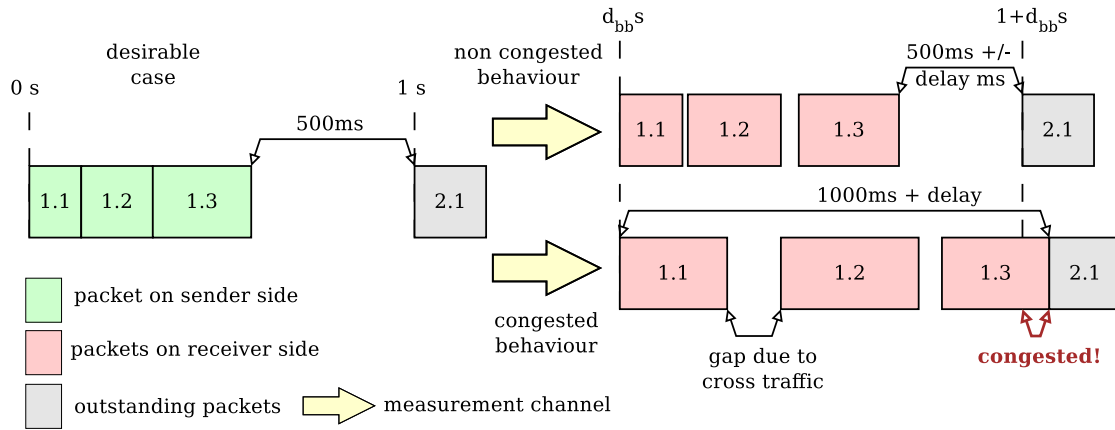


Figure 5.6: Importance of the gap time

The inter-chirp gap t_i^{gap} is defined by the time between the receive time of the last byte of the last packet in a chirp $b_l(n_i)$ and the received time of the first byte of a packet in the new chirp $b_f(1)$. Equation 5.11 holds, when there is no congestion detected:

$$\begin{aligned}
 t_i^{gap} &= t_{i+1}^{rx}(b_f(1)) - t_i^{rx}(b_l(n_i)) \\
 &\approx (t_{i+1}^{rx}(f) - d_{i+1}^{air}(1) - (f-1) \cdot d_{i+1}^{air}(2)) - (t_i^{rx}(l) + (n_i-l) \cdot d_i^{air}(2)) \\
 &= \left(t_{i+1}^{rx}(f) - \frac{sp_1}{A_i} - (f-1) \cdot \frac{sp_{i+1}}{A_i} \right) - \left(t_i^{rx}(l) + (n_i-l) \cdot \frac{sp_i}{A_i} \right) \quad (5.11)
 \end{aligned}$$

When congestion is detected, a new value for t_i^{gap} is needed. d_{OWD} is the OWD of the packet of the safeguard channel. If no packets were received via this channel, d_{OWD} is the one-way delay of the last non-congested packet via the measurement-channel, so it is:

$$\begin{aligned}
 t_i^{gap} &= 0.1 + 3 \cdot d_{OWD} \\
 &= 0.1 + 3 \cdot (t_i^{rx}(k) - t_i^{tx}(k)) \quad (5.12)
 \end{aligned}$$

Equation 5.12 is based on a current OWD, because t_i^{gap} needs to be large enough, therefore a further packet could be send within current t_i^{sleep} . This additional packet will be send due to an additional feedback (see Section 5.5.6). Equation 5.11 contains the current estimated data rate A_i , because A_{i+1} can not be estimated after the first packet of chirp $i+1$ was received, therefore the data rate A_i will be used to approximate d_{i+1}^{air} . If A_i is used, there are two cases:

- $A_{i+1} \geq A_i$ does not harm the calculation, because an overestimated inter-chirp gap time is better than underestimated on. The reason is that a greater value of t_i^{gap} leads to a greater value of t_{i+1}^{sleep} , therefore the sender suspends longer than necessary. Otherwise he would send packets again, when the congestion could not be reduced completely.
- When $A_{i+1} < A_i$ holds, the first term of Equation 5.11 will be underestimated. But this will be handled when a second packet will be received, so that the estimation becomes more accurate. In the worst case there will be no second packet and the detector relies on packets out of chirp $i+2$.

Additionally, the data rate of chirp i could not be calculated, when less or equal one packet of chirp i was received, then the data rate of the previous chirp will be used. When $1 = f$ does not hold, the transmit time stamp was piggybacked and the receive time stamp can be estimated. If $n_i = l$ does not hold, the time stamp will be estimated as shown in Equation 5.7.

If all components are added, current time to sleep can be defined as:

$$\begin{aligned}
 t_{i+1}^{sleep} &= t_{i+2}^{etx}(1) && -t_{i+1}^{rx}(k) \\
 &\stackrel{5.10}{=} t_i^{gap} - d_{bb} + t_{i+1}^{erx}(n_{i+1}) && -t_{i+1}^{rx}(k) \\
 &\stackrel{5.7}{=} t_i^{gap} - d_{bb} + (n_{i+1} - k) \cdot \frac{sp_{i+1}}{A_i} + t_{i+1}^{rx}(k) - t_{i+1}^{rx}(k) \\
 &= t_i^{gap} - d_{bb} + (n_{i+1} - k) \cdot \frac{sp_{i+1}}{A_i} && (5.13)
 \end{aligned}$$

After congestion is detected, mechanism in Figure 5.7 will be triggered, whereat is does not matter, whether client or server detects the congestion. In this figure, terminal 1 detects congestion. This terminal sends a safeguard-packet to notify terminal 2 that congestion occurred. Additionally, the current values of t_{i+1}^{sleep} and the packet id will be set in the backchannel, because the safeguard is optional. It is important that a safeguard message will be send only once per

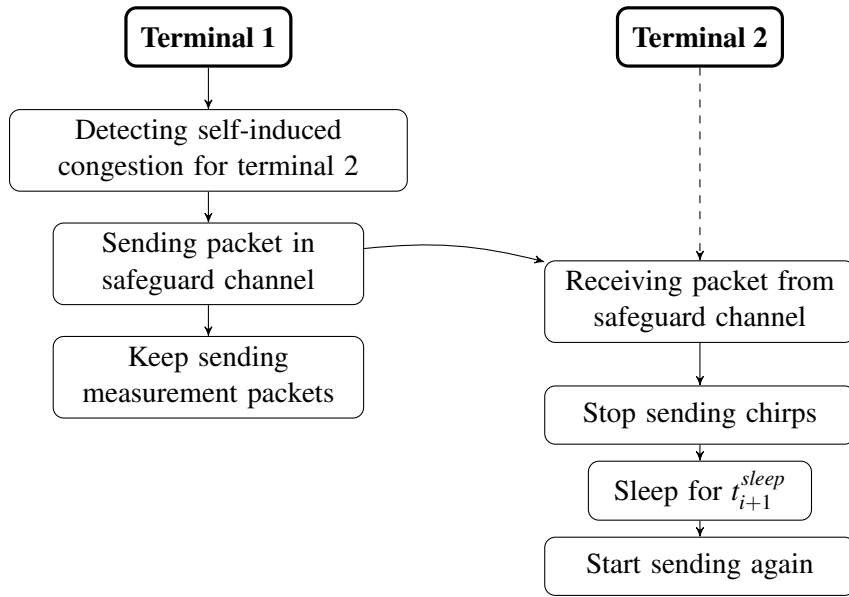


Figure 5.7: Flow chart of the logic after detecting self-induced congestion

congestion. If congestion is detected for another packet, the new value of t_{i+1}^{sleep} will be compared to the old one. If the new calculated value is greater equals the current one, but the previous time to sleep is not over yet, no new message will be send. Hence, the value of t_{i+1}^{sleep} is smaller than the current, a message will be send under certain circumstances, which will be mentioned in the Section 5.5.6. After terminal 2 received the safeguard-packet, he suspends and stops his countdown-timer. Then he will adjust the sleep time from delay of the safeguard channel and sleep until t_{i+1}^{sleep} seconds have passed (see Equation 5.13). Only then the terminal starts sending

again. Because the terminal does not send packets for the last t_{i+1}^{sleep} seconds, the received piggybacked information can be out of date. Maybe this will induce a second phase of congestion, but this cannot be avoided due to missing information about the network. The network is seen as a black box and therefore the terminals have no information for example about other terminals and the amount of cross traffic flows. More details about this phase and a proof of concept is presented in Section 6.1.5.3.

5.5.6 Positive and Negative Feedback

After inter-chirp congestion could be detected, more packets of the current chirp could be received. If for one of these packets the new value of t_{i+2}^{etx} differs from the old estimated value, *positive* or *negative feedback* should be send. To be more precise, it will be checked, how the new value of t_{i+2}^{etx} – denoted as *new* t_{i+2}^{etx} – and the time, the other terminal has to sleep, are distinguishing. Figure 5.8 shows the logic of the feedback-mechanism, whereat t^{now} is the current time and t^{old} denotes the time stamp, when the last safeguard-packet was send due to congestion. First, it is checked, whether the other terminal is already in a congestion phase. If it is “sleeping”, the new value of $t_{i+2}^{etx}(1)$ is compared with the old estimated transmit time stamp of next chirp. Thus a positive (earlier time stamp) or negative (later time stamp) can be determined. Both feedbacks will be filtered by a low-pass filter. This means, only values which differ by 10 % and at least $2 \cdot d_{OWD}$ are accepted. This prevents oscillations due to small values. With the negative feedback, only the current time to sleep has to be set. Whereat at the positive feedback, the countdown-timer (see Section 4.5) will be restarted with a new time, denoted by t^{ct} , and t_{i+1}^{sleep} is assigned to *new* t_{i+1}^{sleep} . Due to safety, a safeguard-packet will be send three times per second, if the other terminals sleep time is greater than $3 \cdot d_{OWD}$ seconds yet and if the other terminal is currently sleeping. These provisions should secure against packet loss in the safeguard channel. Additionally a second countdown-timer is running for half a second plus a buffer of 100 milliseconds. This timer is an additional safety feature and protects against fast changing network characteristics which are not caught by the feedback-mechanism.

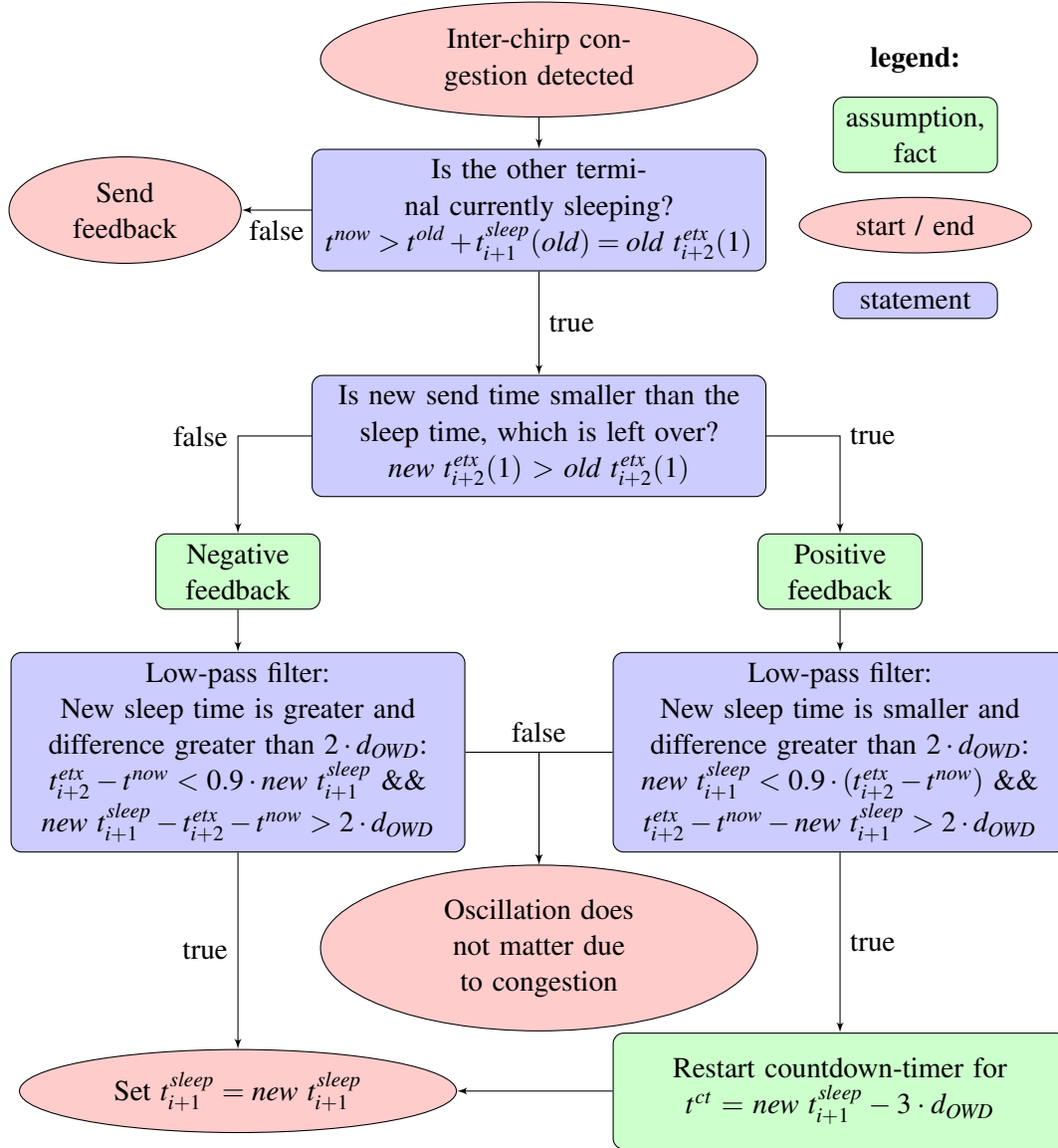


Figure 5.8: Logic of the feedback-mechanism for the sleep time on congestion

Chapter 6

Experiments

This chapter is about testing the novel measurement algorithm in a testbed with laboratory conditions as well as in real-life. Therefore, Section 6.1 is about measurements in the testbed is about the quality of different traffic generator. Additionally default parameters for the RMF and measurement algorithm are mentioned. Further Section 6.2 describes measurements with a stationary client as well as with a moving one.

6.1 Measurements with Laboratory Conditions

Experiments have been done in a testbed with laboratory conditions. Therefore, Section 6.1.1 is about the used hardware. Section 6.1.2 briefly evaluates traffic generators for cross traffic. Next section deals with input parameters of the new measurement algorithm, whereat Section 6.1.4 describes the generation of a chirp. Last Section is about the results of measurements in the testbed.

6.1.1 Hardware

The testbed consisted out of seven computers, whereby specification of the hardware can be seen in Table 6.1. Network setup is shown in Figure 6.1 and the computers are assigned as follows:

- Amy serves as client,
- Fry is the server,
- Farnsworth and Bender are routers and

- Kiff, Leela as well as Nibbler will generate cross-traffic.

Processor	Intel i3-2100
Cores	2
Frequency	3.1 GHz
Main Memory	4 GB
Network Interfaces	10/100/1000 Mbps
Operating System	Debian 7.0
RMF Version	58842

Table 6.1: Hardware of the testbed

The link from Amy to Fry, routed by Farnsworth will be the control- and measurement-channel, whereby the link routed by Bender will be the safeguard channel. Unfortunately, all computers in the testbed are running on Debian. Fortunately the recent GNU-C-Library could be installed with experimental sources on Amy and Fry (see Section 3.4). The maximal available data rate of all links was curbed by *ethtool* (see [Mil13]). With the help of the Iperf (see [Kul14]) this setting was verified. The results of Iperf are near to 10 megabit and can be seen in Listings A.1 and A.2.

6.1.2 Evaluation of Traffic Generators

Section 3.5 introduced several traffic generators. In the following it is explained, why Brute, D-ITG and Iperf are used for traffic generation. Additionally, the generated cross-traffic are analyzed with data rates and time periods out of Tabular 6.2. The test scripts are listed in Appendix A.3.2. Every cross-traffic in this section is generated by Leela and routed by Farnsworth towards Fry. With *ifstat* (see [Rou14]) the interface status has been logged. Ifstats gathers statistics from the kernel internal counters. The CPU frequency is set to the maximum frequency due to the performance governor, because for example Brute exploits the time stamp counter (TSC) register for achieving best temporal accuracy. Therefore, *cpufreq-utils* (see [BD14]) has been used. This script is in listing A.3.1.

Brute outperforms MGEN and RUDE with throughput versus frame length (see [BGPS05]). Additionally, it was used in [Lan13] and worked reliably. For better evaluating of Brute, a second application-level traffic generator should be tested. Unfortunately, KUTE only works with Linux up to 2.6 (see [ZKA05]), but the testbed uses Linux 3.2.

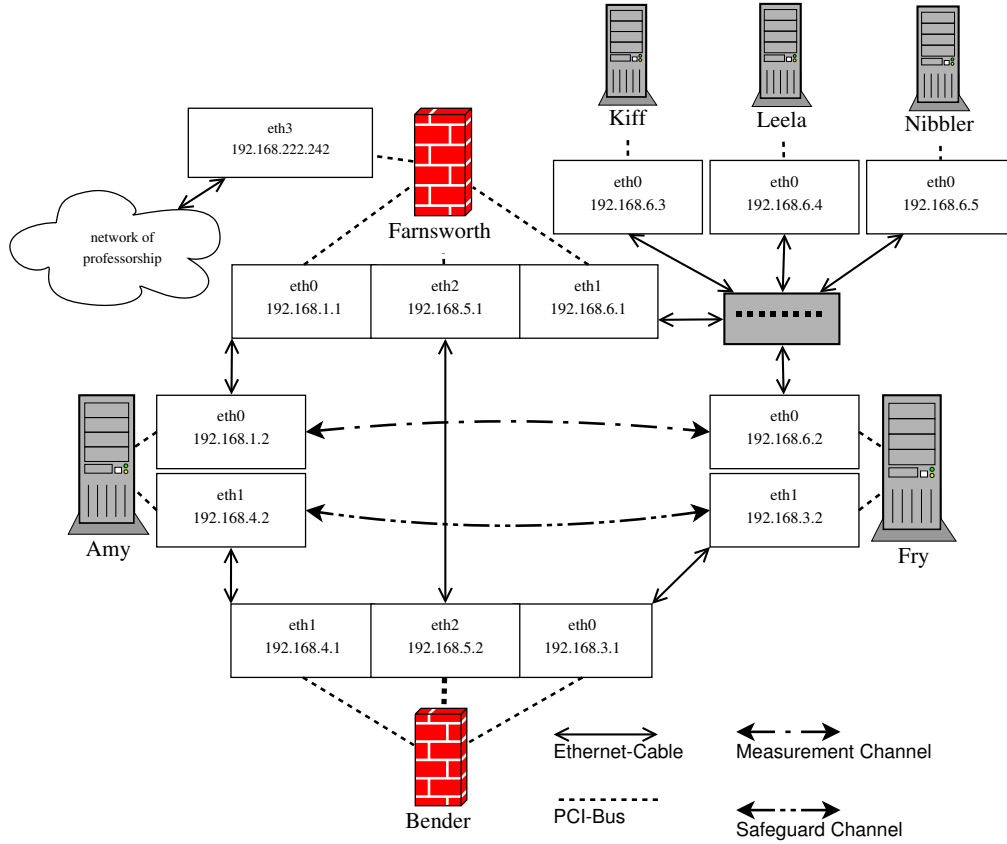


Figure 6.1: Testbed for measurements

D-ITG clearly achieves the best available packet-rate and bit-rate compared to MGEN and RUDE in [BDP10]. But it should be noted, that the evaluation observes packet-rates from 0.6×10^5 to 1.6×10^5 and bit-rates from 3.5×10^5 to 10.6×10^5 , whereby the bit rate in this thesis is shaped to $10\text{Mbit/s} = 10 \times 10^6\text{bit/s}$. However, it is clear that D-ITG remains better than MGEN and RUDE with the aid of the figures in [BDP10].

Iperf is a UNIX standard tool for years. With Iperf the current available data rate of a link can be estimated fast and easy as well as traffic can be generated. Iperf was used in server mode, because every last packet of each measurement needs to be acknowledged, otherwise there will be an additional delay due to a missing acknowledgement

In the following the generated cross-traffic of Brute, D-ITG and Iperf is analyzed. Data rate and periods of time are defined in Table 6.2, where for example the cross-traffic is up to 200KByte/s between 5 and 15 seconds. Brute and D-ITG had a rate of 1000 frames per second, whereby this

options was not available in Iperf. There only the maximal data rate can be set and 1540 Byte Ethernet-packets are send. Scripts are listed in Appendix A.3.2. Obviously Brute is the most appropriate traffic generator, because D-ITG has got deviations downwards, when the data rate changes. Iperf overestimates during higher throughput's due to a base of 1024. Brute is optimal, because it generates one traffic flow with different properties, whereby D-ITG and Iperf have to be restarted for changing properties. This leads to a stitching on the x-axes over time.

Time (sec)	0	5	15	25	35	45	55	65	75
Packet Data Length (KByte)	0	200	400	600	800	1000	1200	600	0

Table 6.2: Data rate of cross-traffic for Brute and D-ITG dependent on time

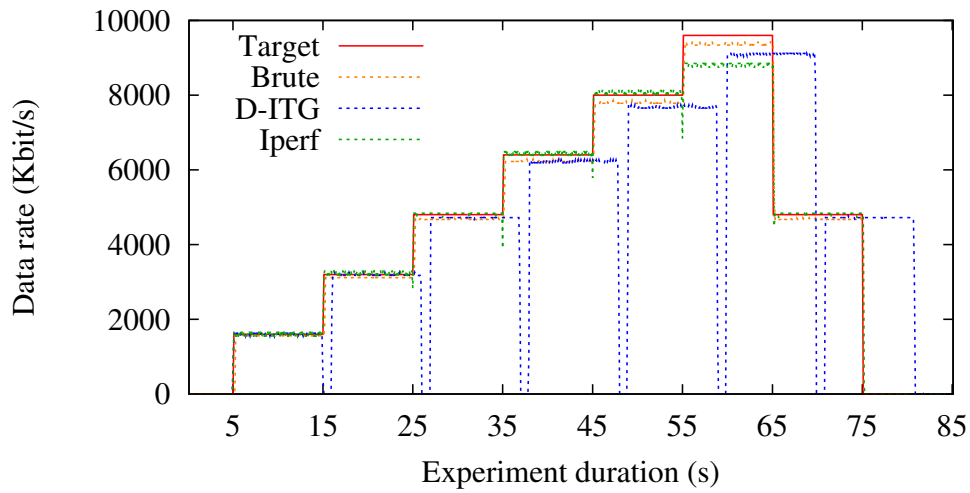


Figure 6.2: Cross-traffic by different generators

6.1.2.1 Traffic Shapping

As an alternative towards traffic generators, traffic shaper like *trickle*, *wondershaper* or *tc* could be used ([Eri14], [Hub14b] and [Hub14a]). Unfortunately, *trickle* only handles TCP-connections. *Wondershaper* is a wrapper for *tc* and adjusts data rates through in- and decreasing drop rates. This behaviour is undesirable, because packets should only be dropped, when a network queue is full. Fortunately *tc* can adjust the queuing discipline for different network interfaces with the help of a *token bucket filter*. This filter is a simple queue that only passes packets arriving at a rate which not exceeds rate set by the user. *Tc* is used in Section 6.1.4 for estimating various properties of a chirp.

6.1.3 RMF Input Parameters

This subsection briefly describes input parameters for the new measurement algorithm of the RMF. Table 6.3 shows all parameters as well as default value and a short description.

Parameter	Value	Default	Description
mT_r	0 (off) 1 (on)	0	Boolean value, whether only results are calculated.
mT_a	0 (off) 1 (on)	1	Should network characteristics affect chirp properties?
mT_d	0 (off) 1 (on)	1	Boolean value, whether SICD is used.
mT_s	byte/seconds	16000	Data rate at beginning has to be low to prevent inter chirp SIC. Therefore 16 Kbit/s are chosen.
mT_l	percentage	50	Percentage of channel load in a range of 0 to 100. As described in Section 5.1 it is 50 %.
mT_cs	integer	3	Denotes value for smoothing the results in result calculator (see Section 4.4) for. Before this thesis was done, default was three, because of fluctuations. This value was kept as default value, but value of 1 is used in this chapter.
mT_cc	integer	25	Amount of packets, after which the chirp duration will be calculated due to congestion detection. The value of 25 was set as default due to CPU load of the Alix-Board.
mT_clu	integer	1000	Maximum chirp length for the uplink. The default value is 1000 due to tests under laboratory conditions.
mT_cld	integer	1000	Analog to mT_clu, but for the downlink.
mT_mdu	byte/seconds	1000000	Default data rate for the uplink is 10 Mbit/s due to tests under laboratory conditions, where the link were shaped to 10 Mbit/s.
mT_mdd	byte/seconds	1000000	Analog to mT_mdu, but for the downlink.
mT_gc	0 (off) 1 (on)	0	Client will use the GPS-helper (see Section 4.2), whereat root privileges are needed. By default this is disabled.
mT_gs	0 (off) 1 (on)	0	Analog to mT_gc, but for the server
mT_gcl	0 (off) 1 (on)	0	Client will log GPS-data with gpspipe, whereat root privileges are needed. By default this is disabled.
mT_gsl	0 (off) 1 (on)	0	Analog to mT_gcl, but for the server.
mT_tc	integer	2947	Gpsd TCP port of client. Default value bases on gpsd.
mT_ts	integer	2947	Analog to mT_tc, but for the server.
mT_sc	string	/dev/ttyS0	Gpsd serial port of client. Default value bases on gpsd.
mT_ss	string	/dev/ttyS0	Analog to mT_sc, but for the server.

Table 6.3: Input parameter for the new measurement method

6.1.4 Estimating Chirp Length and Packet Size

Chirp lengths and packet sizes have to be calculated due to adjust the estimated data rate of the receiving terminal. Otherwise the terminal would produce self induced inter-chirp congestion. Indeed the calculation of both parameters is not trivial, because there is a trade off between maximal data rate of data and robustness. As is well known smaller packets lead to higher overhead, more calculating time and less data rate of data, but loss does not have a major impact on the data rate. However, if packets get bigger, loss has a huge negative effect, but the overhead can get very small. Therefore, two approaches were tested. Firstly, data rate dependent on different packet sizes and fixed chirp length and secondly, data rate dependent on different chirp lengths and fixed packet sized.

6.1.4.1 Chirp properties based on different packet sizes

Firstly the maximal used bandwidth with different packet sized will be estimated. Table 6.4 denotes different packet size for different time intervals, whereby for example in the interval of [23,33) seconds all packets have a size of 1400 bytes.

Time (sec)	3	13	23	33	...	143	153
Packet Size (kBit)	500	1500	1400	1300	...	200	100

Table 6.4: Packet sizes in dependent of time

The results for a measurement with the properties of Table 6.4 are shown in Figure 6.3. The results are from the servers point of view, whereby the clients download is equal. Every coloured line represents a single measurement and the link was shaped with tc. Loss has no unknown side effects on this measurement and can be neglected. As can be seen, the first two to three seconds are inconsequential, because of a slow-start (see Section 5.2). Obviously there is no direct correlation between packet size and estimated data rate on a perfect link in Ethernet. In this thesis a perfect link is known as connection between sender and receiver without cross-traffic or unknown loss. It should be noted, that there are deviations starting from second 133 on the 10Mbit/s link. This is due to chirp length and the time for calculations, which arises from the maximal 6250 packets⁹. As a result for the final calculation, the chirp length has to be limited.

⁹ $10000\text{KBit} \cdot 0.5 \div 100\text{Byte}/\text{packet} = 6250\text{packets}$

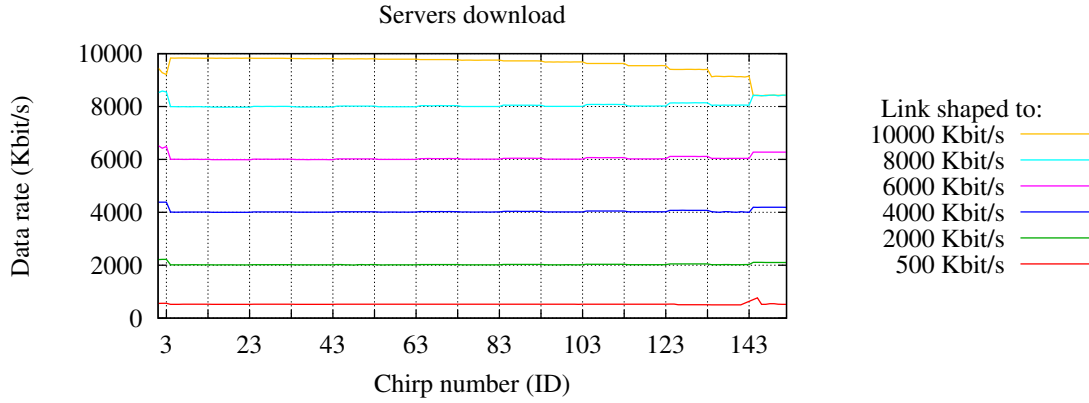


Figure 6.3: Data rate dependent on different packet sizes

6.1.4.2 Chirp properties based on different chirp lengths

The second approach calculates the packet size as result of chirp length. Therefore, the packet size can be calculated as division of data rate through chirp length multiplied with the channel utilization in seconds:

$$sp_i = \frac{A_i \cdot 0.5}{n_i} \text{ bytes} \quad (6.1)$$

Packet size has to be in $[100, 1500]$, because this is the interval of double the dynamic and fixed payload as well as the maximum transmission unit in Ethernet (see [MD90]). If this interval can not be hold chirp length will be in- or decreased. Figure 6.4 displays the results for different chirp length, whereby the link was shaped with tc as denoted in Table 6.5. As can be seen, there is no direct correlation between chirp length and estimated data rate on a perfect link in Ethernet, too.

Time (sec)	13	23	33	...	93	103	113
Data rate (kBit)	10000	9000	8000	...	2000	1000	500

Table 6.5: Data rate in dependent of time

6.1.4.3 Chirp properties based on different chirp lengths and different packet sizes

The results of both approaches in Section 6.1.4.1 and 6.1.4.2 are not surprisingly, because all tests were done in a Ethernet network with perfect links. Therefore, the packet size of chirp i will be calculated like in Figure 6.5.

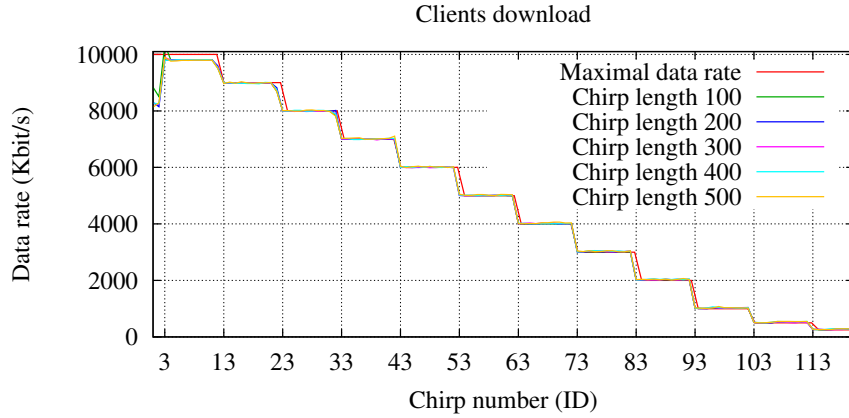


Figure 6.4: Data rate dependent on different chirp lengths

The chirp should have a length of 400 packets, whereby the packet size will be calculated like in Equation 6.1. Additionally, the packet size has to be in the range from 100 (double the size of fixed and dynamic payload) to 1500 (maximal transfer unit in Ethernet). If this limit is exceeded, the packets will have the size of the biggest/smallest value and the chirp length will be modified. The maximal chirp length is 1000. This is a default value out of the header of *networkThreadStructs.h* in the RMF.

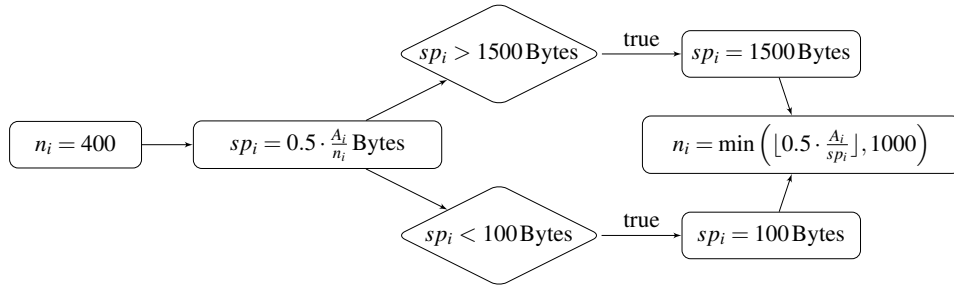


Figure 6.5: Calculation of packet size and chirp length

6.1.4.4 Channel Utilization

As mentioned in Section 5.1, the measurement algorithm tries to achieve a channel utilization of 100 % for 500 milliseconds per second. However, the novel measurement algorithm and SICD can handle different values for the channel utilization, which can be set by *-mT_l* (see Section 6.1.3). This possibility should be used in an extended version of this algorithm only, because the

utilization directly influences the possibility of SIC. With a utilization of 50 % no SIC will occur until the available data rate is less than half of the expected rate.

6.1.5 Measurements

A variety of measurements under laboratory conditions in a wired testbed were done. This measurement will cover in the following subsections. During all measurements the RMF parameters were set like depicted in Table 6.6 and the plots show unfiltered and non smoothed data.

Option	Value
Binding of Process	1 (enabled)
LogLevel	1 (normal)
Process Scheduling	fifo
Deactivate swapping of Memory	1 (enabled)
Performance Scaling Governor	1 (activated)
Measured Link	both (up- and downlink)
Client Control Channel Address	192.168.1.2
Client Measurement Channel Address	192.168.1.2
Client Safeguard Channel Address	192.168.4.2
Client Safeguard Interface	eth1
Client Safeguard Default Gateway	192.168.1.1
Server Control Channel Address	192.168.6.2
Server Measurement Channel Address	192.168.6.2
Server Safeguard Channel Address	192.168.3.2
Count of Smoothing	1

Table 6.6: Options for measurements in the testbed

6.1.5.1 Measurement without loss and cross-traffic

Measurement in Figure 6.6 was done between Amy and Fry. At the beginning of the measurement a “slow-start” can be seen. This was already mentioned in Section 6.1.2.1. The results only deviating $\approx 2.32\%$ for the uplink and $\approx 2.56\%$ for the downside of the client. The underestimation is better than a overestimation, because then the risk of inter-chirp congestion is smaller. The difference of deviation is due to measuring fluctuations. Summarized, these results are very good for a perfect link.

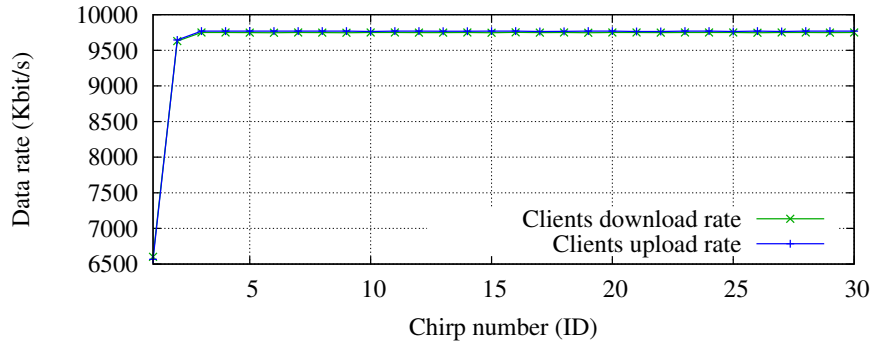


Figure 6.6: Measurement results for an unencumbered link

6.1.5.2 Measurement with loss

Previous measurement was done with perfect link, whereby now the algorithm is tested on a link with loss. It is expected, that $x\%$ of lost packets affects on $x\%$ less estimated data rate, because less packets will be received in the same amount of time (on condition, that the last packet is received). Table 6.7 displays the loss on the down- and uplink based on time, where for example the links were affected with 20 % loss in $[15, 25)$ seconds. The emulation of loss can be done with the network emulator *NetEm* for tc (see [Hem14]), which adds packets loss to an outgoing interface in the network. Unfortunately, *NetEm* is not exact, thus *iptables* could be used (see [RN14]), but even *iptables* has a probability module and is not exact. Therefore, *NetEm* was used, because it sets the loss probability to the network queues itself.

Time (sec)	5	15	...	85	95	105	110
Loss (Percent)	10	20	...	80	90	95	0

Table 6.7: Periods of loss for testing

Figure 6.7 shows a measurement with the new developed algorithm and with variable loss, which is displayed in Table 6.7. The behaviour of the client's upload is equal. While the data rates are plotted versus the left vertical axis, the loss rates are plotted versus the right vertical axis. Ideally the red and green line would match exactly. The yellow graph depicts the difference of the measured available data rate in respect to the estimated available data rate. It is clearly visible that if loss increases up to 20 %, the estimated data rate will decrease up to 20 %, because 20 % less packets are received. At 107th second it can be seen, that the loss rate is set to zero and thus the data rate is increasing to the maximum like in the "slow-start".

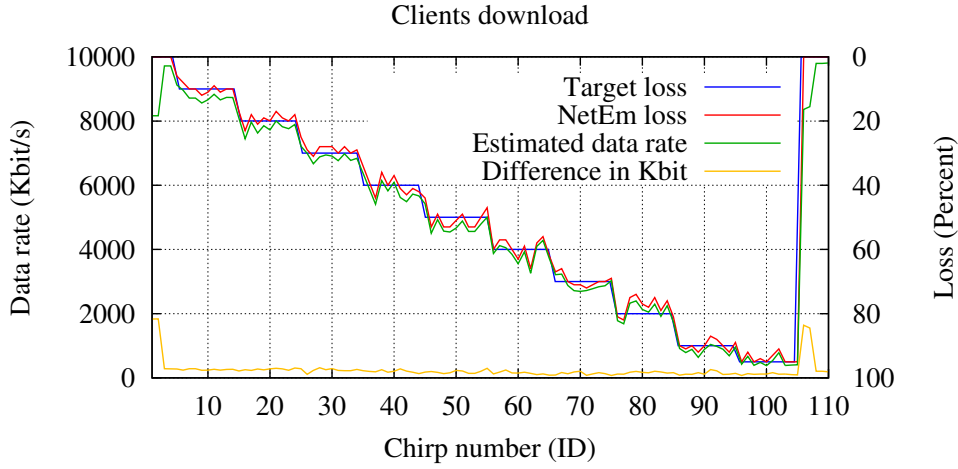


Figure 6.7: Measurement results for a link with loss

6.1.5.3 Measurement with cross-traffic

After the new measurement algorithm was evaluated on a link with loss, measurements with cross-traffic will be evaluated. For the generation of cross-traffic it is important to know, that the kernel scheduler is fair and implements a packet FIFO-queue. Hence, the channel capacity will be divided fair on the basis of flow count per terminal. Therefore, the test cases contain different amount of flows, which were generated by Leela, Kiff and Nibbler. These flows will influence the client's upload. The plotted cross-traffic is always the specification and not the real value, whereby blue lines denote the value of a fair division based on the FIFO-queues. This subsection contains following measurement:

- Test 1: with one cross-traffic, so that there will be no congestion.
- Test 2: with three cross-traffic flows, but every flow has a delay in time to avoid congestion.
- Test 3: with two cross-traffic flows at the same time, so that there will be congestion.
- Test 4: with two cross-traffic flows, which will vary in time and with congestion.

Cross-traffic test 1: Figure 6.8 shows a measurement with one increasing cross-traffic flow, which was generated by Leela with Brute. The cross-traffic flow had 1000 packets per second with increasing packet size every ten seconds. The fairness of the kernel respectively Ethernet can be seen starting at 70th second respectively chirp number. As mentioned in Section 5.1 the new measurement method aspires a channel utilization of 100 % for 50 % of time. Therefore the the channel is divided fair and the results of the measurement algorithm are predictable.

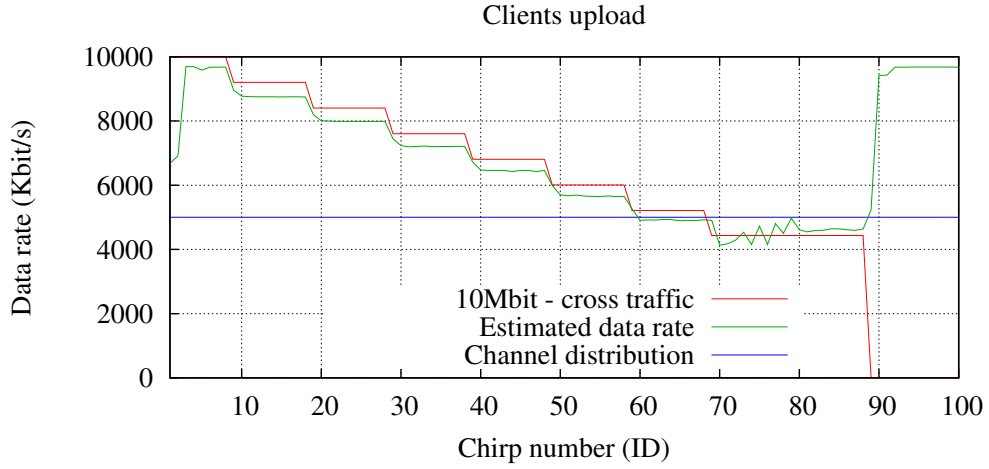


Figure 6.8: Measurement results with one cross-traffic flow, without congestion

Cross-traffic test 2: The second cross-traffic test was done with three flows, generated by Leela, Kiff and Nibbler. Each flow has a size of 1000 packets per second, a packet length of 1000 byte and a duration of 60 seconds. The first flow started after 10 s, the next at 30 s and the last at 50 s. Each flow lasted for 60 s and receives about $1/\text{count of flows}$ of the bandwidth. The only risk of congestion is the start of the first flow, because the available data rate is reduced by one half. Figure 6.9 shows the measurement with non smoothed results. The prongs of the estimated bandwidth are based on the kernel's scheduler.

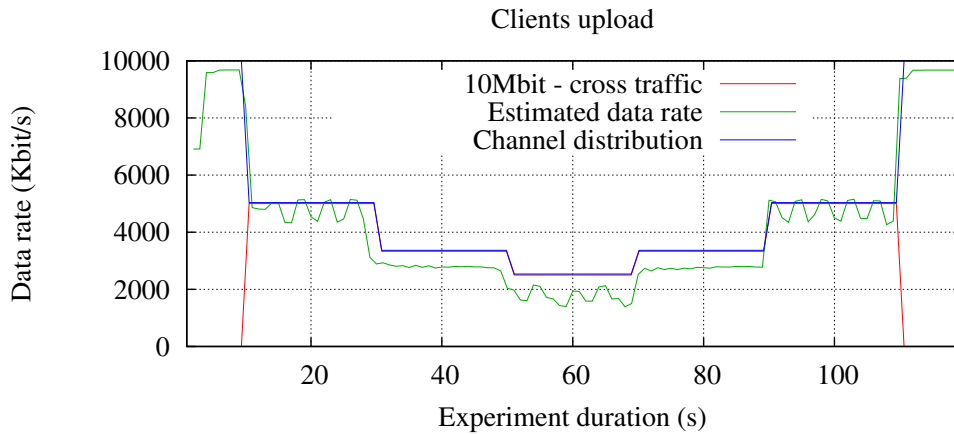


Figure 6.9: Measurement results with three cross-traffic flows, without congestion

Cross-traffic test 3: The third cross-traffic test will lead to congestion. Therefore, two cross-traffic flows with CBR will be generated after 9th second for 10 seconds. This will lead to an decrease of the data rate of approximately 66 %. Due to this congestion, the server notifies the client via the safeguard channel. The received values of t^{sleep} of the safeguard packets are displayed in the lower part of Figure 6.10. Theoretically the sleep time based on the 10th chirp is equal 2 seconds ($t_{10}^{sleep} = 2$), because the available data rate is divided by 1 : 2. In practice, there is an inaccuracy due to start time of the cross-traffic. Therefore the first time to sleep is about 1.6seconds. As can be seen, this value is corrected due to negative feedback. This feedback is based on the fact that the congestion declines slower as anticipated. It is important to note, that chirps can not be send once per second, when congestion occurs. The reason is, that t^{sleep} has influence on the sending time of the next chirp. Therefore, the x-axis is not labeled with “Chirp number (ID)” but with “Experiment duration (s)”.



Figure 6.10: Measurement results with two cross-traffic flows and with congestion

Cross-traffic test 4: Last test with controlled cross-traffic will be done with several congestion phases. It is expected, that the new algorithm should adapt different phases of congestion due to exact sleep times. Two flows will be generated at the same moment for same amount of time. In this test both flows were generated for 0.5, 1, 2, ... and 8 seconds, every time with a 5 second break except the first interval. Figure 6.11 shows the estimated data rate, cross traffic and the values of t^{sleep} due to SICD. As can be seen, several safeguard-packets are received. The reason for multiple safeguard packets per congestion is the same as in Figure 6.10: t^{sleep} does not considers the duration of cross traffic and feedback is send. Additionally the sending rate of the

Amy changes due to cross-traffic. It follows, that Amy calculates the available data rate suitable. Peaks below 33 % are related to the routers fifo queue. An evaluation of delays will be done in a more complex case in the next section.

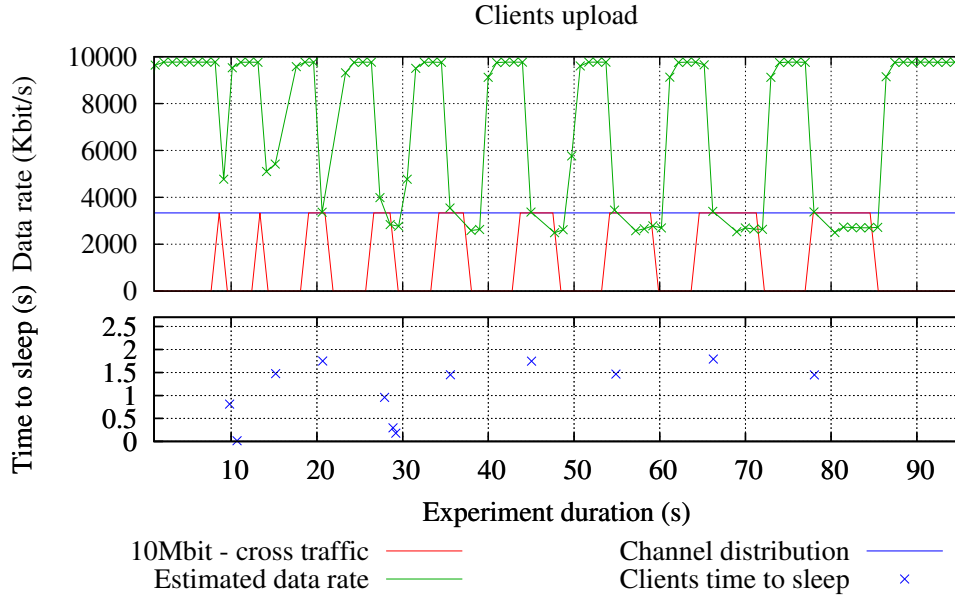


Figure 6.11: Measurement results with two cross-traffic flows and several congestion periods

6.1.5.4 Measurement with Random Traffic

The last measurements in the testbed will be done with random traffic. Actual Ethernet traffic is self-similar and bursty (see [LTWW93]), but none of the traffic generators, mentioned in Section 6.1.2, are able to generate this kind of traffic. In view of the mentioned traffic generators, D-ITG is able to generate UDP flows with various inter-departure times, but none of the generated flows is bursty or self-similar. Therefore, Brute was used again simulating constant bit flows. Figure 6.12 shows a measurement with one random cross-traffic flow, whereby there were two random flows for the measurement in Figure 6.13. The cross traffic was generated for the client's upload, therefore the download link was not influenced. The red line denotes bandwidth minus cross traffic, whereby the cross traffic was measured with ifstat and a resolution of ten measurements per second, whereby nothing below of the channel distribution is plotted. The properties of the random flows were generated with a random number generator¹⁰, which used the *Mersenne*

¹⁰See <http://rechneronline.de/zufallszahlen/withMersenneTwister>.

Twister (see [MN98]). With this generator numbers for the duration, packet rate and size were generated, whereby:

- Duration is in an interval of 0.0 to 5.0, rounded to the first decimal place. Thus the times are big enough, so that the maximal available bandwidth can be estimated and small enough, so that the changes of the cross traffic influences the measurement.
- Packet rate is in an interval of 100 to 1000, rounded off to the powers of hundred. Thereby the range is between the maximal packet rate of the RMF (see Section 6.1.4.3) and a lower limit.
- Packet size is in an interval of 200 to 1500, rounded off to the powers of hundred. Thus the size is from the interval of twice the minimal packet size and maximal transfer unit in Ethernet.

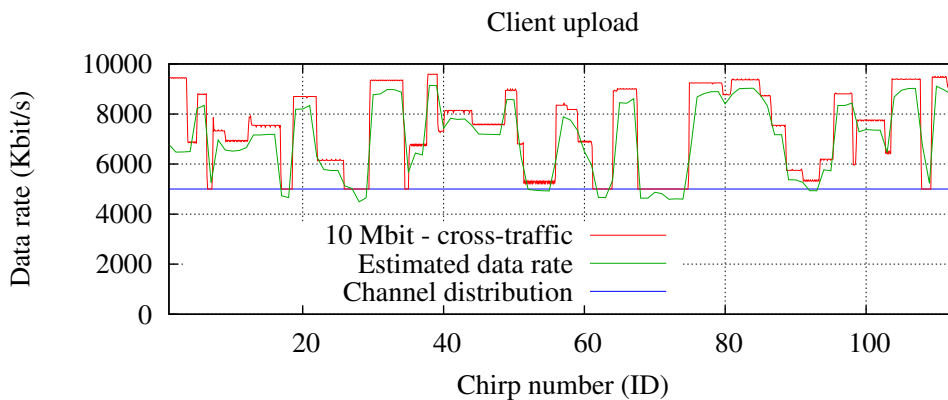
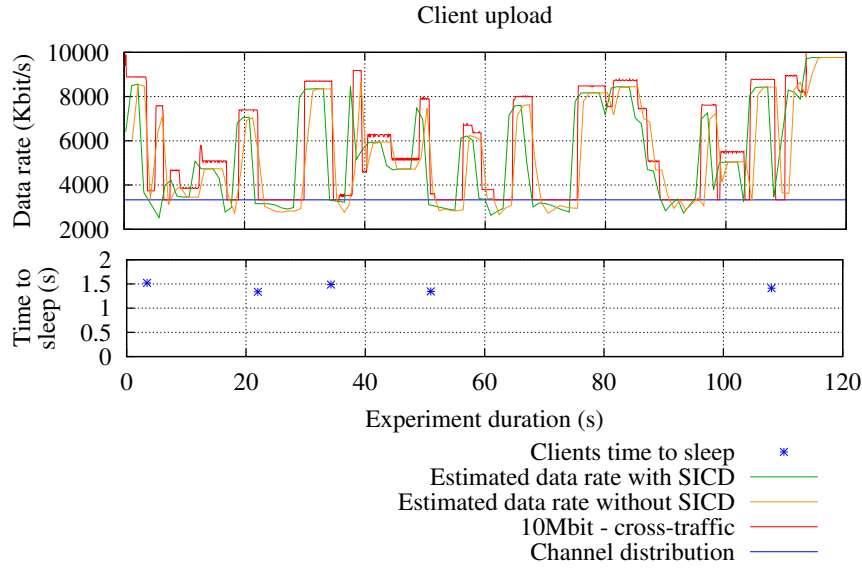


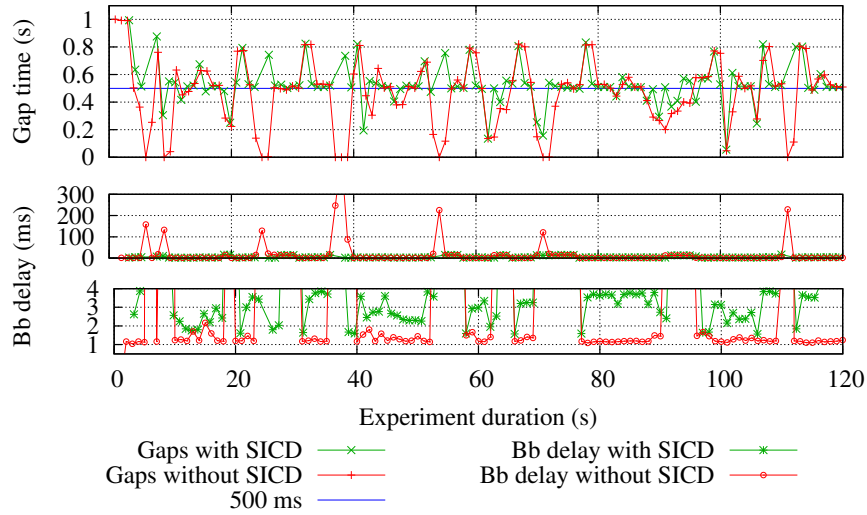
Figure 6.12: Measurement results with one random cross-traffic flow

In Figure 6.12 can be seen, that the maximal generated cross traffic cannot exceed the limit of 50 % channel distribution, because only one flow was generated. Therefore, the biggest drop of the available data rate only can be up to 50 % of the recently estimated data rate and that is why no congestion occurs. As shown in Figure 6.12, the current data rate was estimated satisfactorily. However, the estimation only is a approximation, which can be seen at 90th seconds. There the cross traffic is like a step function, but the data rate estimates a bent line. This is based on cross-traffic, which changes during the chirps. Thus fast and short changes in the cross traffic or available data rate cannot be noted. This is important because for example in UMTS a radio-frame only has a length of 10ms.

Figure 6.13a displays two measurements, each with two random flows, which have the same properties as the flow in Figure 6.12. Therefore the influences of the cross-traffic flows are much stronger. The first measurement (green line) used the safeguard channel with the new SICD



(a) Estimated data rate and sleep times of the client



(b) Backbone delays and gap times for the client

Figure 6.13: Measurement results and delays with two random cross-traffic flows

and the second measurement (orange line) does not. The differences of both lines is based on congestion. Without SICD the terminal sends chirps without noticing congestion. Therefore congestion can not be reduced and time stamps are delayed. This harms the estimation of the data rate and makes the difference in the data rates. The offset in time is based on the sleep times, when SICD is used. Figure 6.13a is divided in two parts. The upper part follows the same structure as Figure 6.11. Second part shows the sleep times of the client. As can be seen based

on the sleep times, congestion occurred after 5 s, 23 s, 36 s, 52 s, and 109 s. As result, the results with SICD are softer and got less peaks. Especially the data rate in the first phase of congestion (around 5 s) could be estimated better with SICD. For a better comparison Figure 6.13b shows the inter-chirp gap times and backbone delays of the first packet in every chirp. While the non SICD measurement shows clear signs of chirps send back-to-back to each other (increasing backbone delay), the gap time of Formula 5.11 respectively 5.12 holds for the SICD algorithm. Signs for congestions in this figure are zero or almost zero inter-chirp gaps and increasing backbone delays due to the congestion in routers FIFO queues. As a result all measured chirp delays are valid for the SICD measurement. Unfortunately the backbone delay of the first packets of each chirp is higher, when SICD is used. This is due to the calculations of SICD, which influence the setting of receive time stamps set by the RMF.

Backbone delays and inter-chirp gap times for Figure 6.13 are shown in Figure 6.14 as cumulative distribution functions (CDF). The left CDF shows gap times, middle plot is detail of the gap times and the right CDF shows the backbone delays of the first packet of the chirps, whereby the xrange is limited to 30 ms.s The delay in transmission is not interesting, because the only the backbone is influenced by congestion in this measurement. The blue line denotes the aim of a 500 ms gap for non congested chirps, which is reached by approximately 20 respectively 30 to 60 % of the chirps. Gap times are less close to zero with congestion detection respectively the smallest gap has size of 55 ms, which is equals the last non congested gap time (see Formula 5.11 and 5.12). It is becoming clear, that SICD is a good improvement for the measurement in Figure 6.13 and the RMF. Of course the algorithm was tested in an environment with data rates, which changed after a time of of 10 to 100 ms, but due to the resolution of 1 Hz, the results were scarcely surprising.

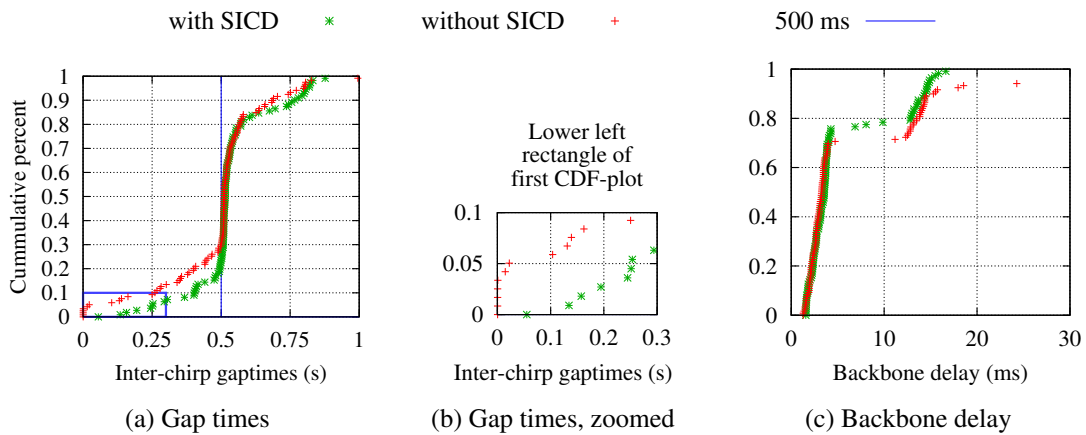


Figure 6.14: CDF of gap time and backbone delay for the upload of the client in Figure 6.13

6.2 Real Life Measurements using Mobile Cellular Networks

Real life measurements were conducted using a mobile client and a stationary server. The measurement server was placed in the housing area of the Heinrich-Heine University's data center and had access to gigabit Ethernet. The NTP settings used for time synchronization and cron job for saving statistics are shown in Listings C.6 and C.8. The technical data of client and server is listed Table 6.8. The server has two gigabit network interfaces using public IP addresses in the same subnet. Therefore, source-based routing is used. All scripts are listed in Appendix C.4.

	Server	Alix-Board
Processor	2 × Intel Xeon CPU 5160	AMD Geode LX800 CPU
Cores	2 × 2	1
Frequency	3.0 GHz	500 MHz
Main Memory	8 GB	256 MB
Network Interfaces	2 × 10/100/1000 Mbps	Sierra Wireless Air Prime MC8790 Huawei UMTS Stick
Operating System	Debian 7.0	Ubuntu 14.04
RMF Version	58868	58868

Table 6.8: Technical data for the server and mobile node

The mobile client uses an Alix-board as its basis. While these boards are build for mobile use, they lack computational power. Table 6.8 shows the hardware, whereby during measurements CPU load is between 30 % and 45 % and RAM usage is approximately up to 100 MByte. As mention in Section 4.3 `gpsd` and NTP were used for time synchronization. Therefore, the `gpsd` configuration of the board can be seen in Listing C.5. Listing C.7 contains the NTP settings of the Alix-board. The UMTS-login was done with `wvdial` (see [Fre14]), whereby the scripts can be seen in Appendix C.1.

In the following two types of measurements under real conditions are done: Firstly, Section 6.2.2 describes measurements with a stationary client. Secondly, the measurement algorithm is tested with a moving client. The results are shown in Section 6.2.3. Input parameters of client and server are shown in Tabular 6.9, whereby $\$1$ is the experiment duration in seconds, $\$2$ denotes IP address used for the measurement- as well as control-channel and $\$3$ is the IP address used for the safeguard-channel. Due to purposes of security the first three octets of the IP address are denoted as $x.y.z$.

Server	-l 1 -P 1 -a 1 -G 1 -L 1 -r server -sc eth0 -sm eth0 -ss eth1
Client	-t \$1 -l 1 -P 2 -M algo -a 1 -G 1 -L 1 -m both -g on -mT_cs 1 -mT_gc 1 -mT_gcl 1 -mT_sc /dev/ttyS4 -r client -F alix3 -CC \$2 -CM \$2 -CS \$3 -SC x.y.z.228 -SM x.y.z.228 -SS x.y.z.229

Table 6.9: Parameters for the server and client for real life measurements

6.2.1 Analyzing Data with KML-Files

To analyze measurement data, a KML-generator¹¹ (keyhole markup language) was written. Figure 6.15 shows a measurement drive in *Google Earth*¹². As can be seen on the left rectangle, marked by “1.”, different files are available. Files with the suffixes “path_down.kml”, “path_up.kml” contain the measurement ride divided in down- and uplink. The legend for the paths can be seen

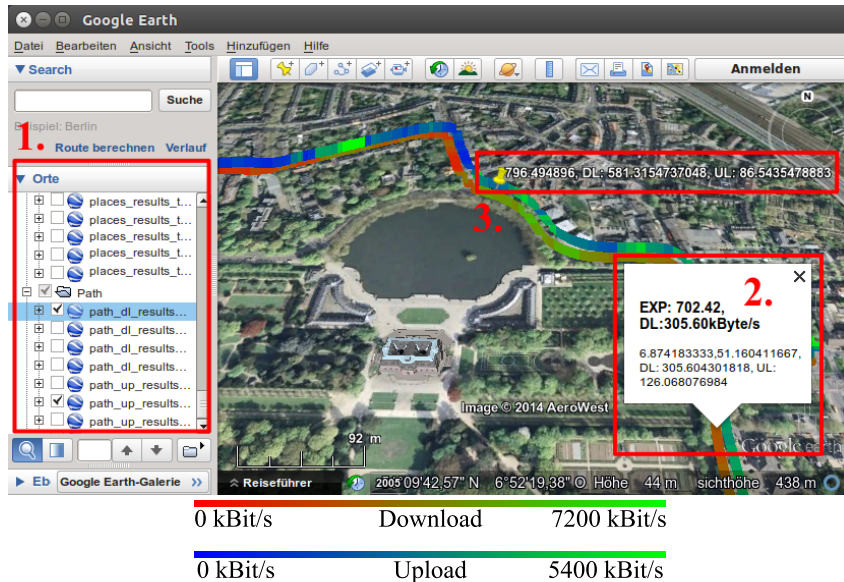


Figure 6.15: Example of a measurement ride in Google Earth

in Figure 6.15. The “places.kml”-files contain the measurement points themselves. These points are GPS-data, linked with the nearest send chirp. The second rectangle displays information about one trace point. As can be seen, the GPS-dates and data rates are saved as the description of the point. The third rectangle, marked by “3.”, shows one example point of the measurement. The KML-generator was written in Python. As input data it expects the folder, client name and maximum downstream and upstream data rates.

¹¹ See <https://developers.google.com/kml/>.

¹² See <http://www.google.com/earth/>.

6.2.2 Mobile cellular network measurement using a stationary client

To test the robustness of the new measurement algorithm, the algorithm was tested with a fixed position. The client, UMTS-antenna and GPS-mouse were placed behind a window at the Heinrich-Heine-University. Antenna and GPS-mouse had a clear view of the sky at a 70 degree angle measured from the ground. Control- and measurement-channel were connected to the mobile cellular network of the provider O2, whereby the safeguard-channels UMTS modem used the mobile cellular network of *Deutsche Telekom*.

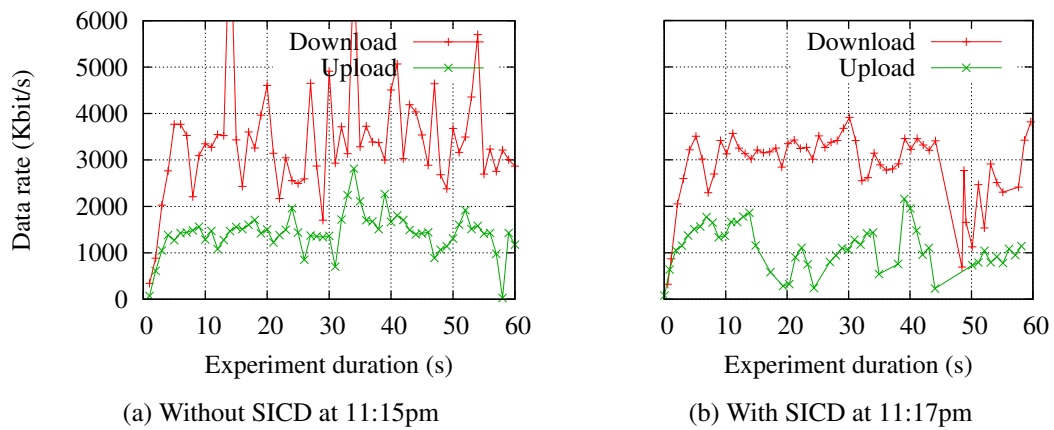


Figure 6.16: Data rates of the client at October 29th

Figures 6.16, 6.17 and 6.18 show various data from a measurements done at October 29th at 11:15pm and 11:17pm for a duration of 60 seconds. Measuring without congestion detection was done at 11:15pm (Figure 6.16a and 6.17), whereby the measurement with detection was done at 11:17pm (Figure 6.16b and 6.18). As can be seen, the estimated data rate is smoother when the new congestion detector is used. This is based on the sleep times, which have influence on the sending time of the chirps. Due to less congestion, less packets are congested and the time stamps are not falsified.

The most important reason for the SICD were wrong delay measurements. The positive effects of SICD are clearly visible when comparing Figure 6.17, showing the delays and gap times of the measurement without SICD, and Figure 6.18, showing the delays with SICD. Here “Bb delay” is short for backbone delay. As shown in Figure 6.16, the data rates are smoother and have less peaks with congestion detection. As can be seen in Figure 6.17, delays of the first packets of a chirp are increased due to starting in a SIC. However the inter-chirp gap time is in between 0.5 s and 1 s. The blue line shows the aim of a 500 ms gap. On the basis of loss, the aim of a 500 ms

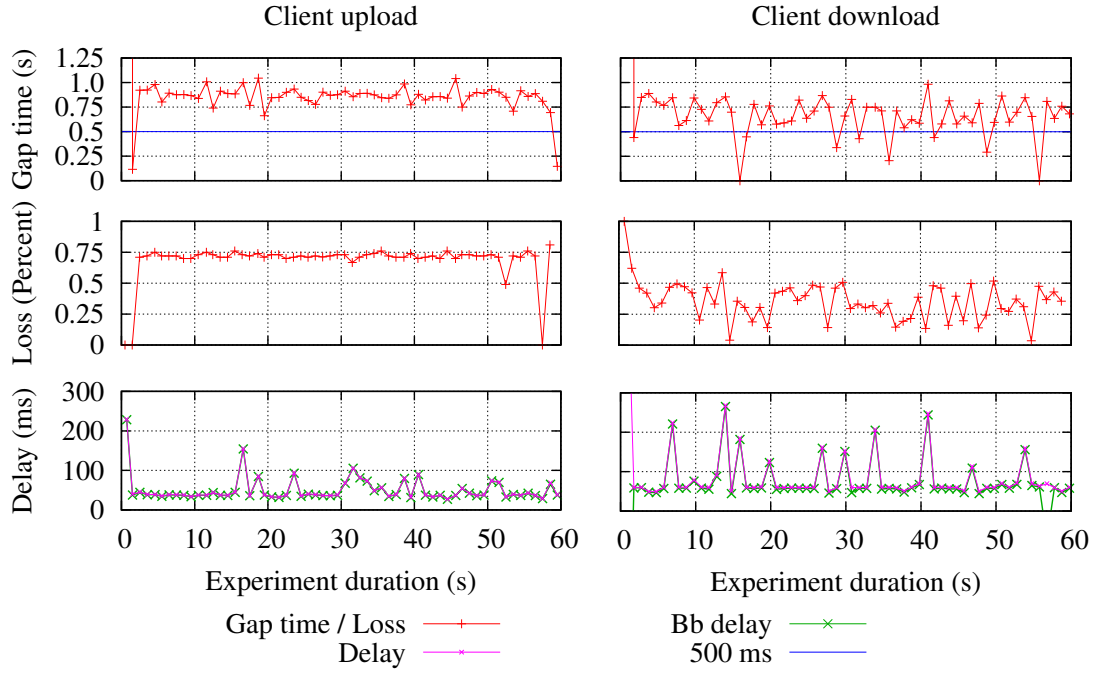


Figure 6.17: Delays and gap times of the measurement at October 29th, 11:15pm, without SICD

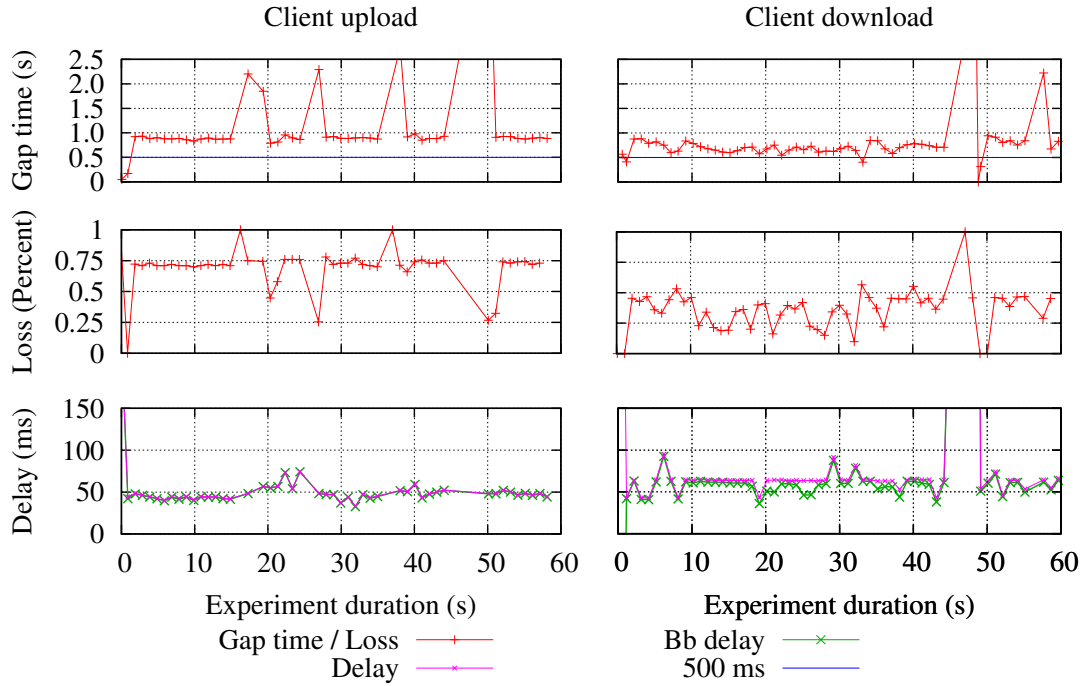


Figure 6.18: Delays and gap times of the measurement at at October 29th, 11:17pm, with SICD

inter-chirp gap could not hold. Unfortunately the tails of the chirps were lost, thus the gaps are increasing. For example, if the the last 50 % of the chirp are lost, the gap will increase up to 75 %. As can be seen, this phenomenon holds for both measurements and can be verified with the logfiles.

With SICD used, the delays are increasing to more than 2 seconds. Normally, this should never happen, but the data reveals lost chirps at these times. If a chirp gets lost, the gap time increases by one second. As can be seen, the difference of the delay of transmission is greater in the down- than in the upload of the client, with the exception of 47th second. There the gap time is greater than the 3 seconds, because the server had an error calculating his sleep time. This bug was fixed later in combination of the feedback mechanism. However, the delay in transmission is not based on an error due to the terminals, but due to congestion in the network of the provider.

The difference of delays between the up- and download could not be reproduced in every measurement which was done in the scope of this thesis. Thus, these fluctuations are due to the assumption that the mobile cellular network is seen as a black box and no information about this box is available for the measurements. For a better representation of the delays, backbone delay and delay of transmission of Figure 6.16 are shown as boxplots in Figure 6.19.

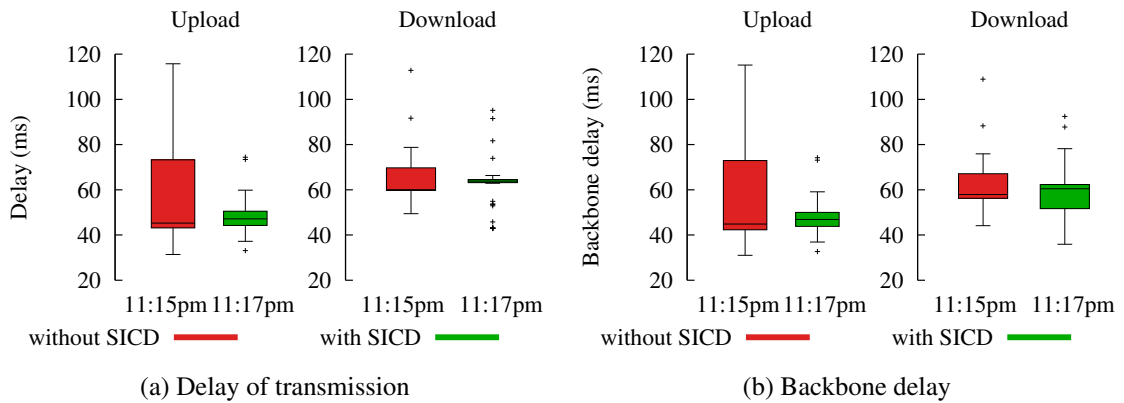


Figure 6.19: Delay and backbone delays of the client at October 29th

In the measurement shown in Figure 6.18, only 57 chirps on the clients downlink and 48 on the clients uplink has been transmitted. Due to the small count of send chirps, the boxplots for the measurement in Figure 6.19 are far from reality, but the improvement by SICD is already visible. Due to the small experiment duration new measurements were conducted at October 30th. They lasted 10 minutes and are shown in Figure 6.20. Two measurements were done without SICD (04:10am and 04:55am), while the other two measurements were done with congestion detection at 04:25am and 05:15am. The client was at the same place as in the previous measurement. The experiments were conducted early in the morning to minimize influences of other users. As can

be seen in Figure 6.20, the range of the box is smaller with SICD used. Whiskers of the boxplots expand from the ends of the box to the most distant point whose delay-value lies within 1.5 times the interquartile range.

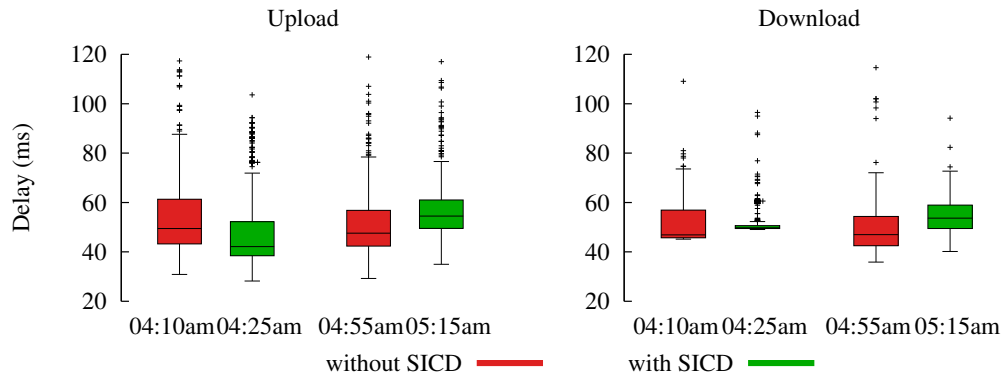


Figure 6.20: Delay of the first packet in chirps of the client at October 31th

6.2.3 Mobile cellular network measurement using a moving client

Measurement rides in this section, were performed on November 4th from 10:15 pm until 11:25 pm in a suburban of Dusseldorf. At this time, few people were around and as a result influences are higher than in the measurements with stationary client at 04:10am at the data center (see Section 6.2.2). The measurements were conducted in the evening, because at this time the repeatability is easier to maintain. The route of the measurement ride can be seen in Figure 6.21, whereby the area of the orange rectangle was particularly conspicuous. It takes approximately 10 minutes to traverse the shown measurement route.

During the measurement ride, the GPS-receiver and the mobile cellular antenna have been placed on top of a car. The UMTS stick was placed directly behind the front pane. The same network as in Section 6.2.2 was used, with just one difference, the connection between a laptop and the server was also hosted by Deutsche Telekom. All channels were hosted by the same providers as in Section 6.2.2. A second connection has been provided by Telekom for watching the status of the server. For this the status messages of the server were disable, so that packets were only send on termination of the RMF. Therefore the extra load on the safeguard channel was very low.

Figure 6.22 displays available data rate and loss rate for this measurement ride. As can be seen, there is higher loss in the download than in the upload. This is due to the air interface. Down-

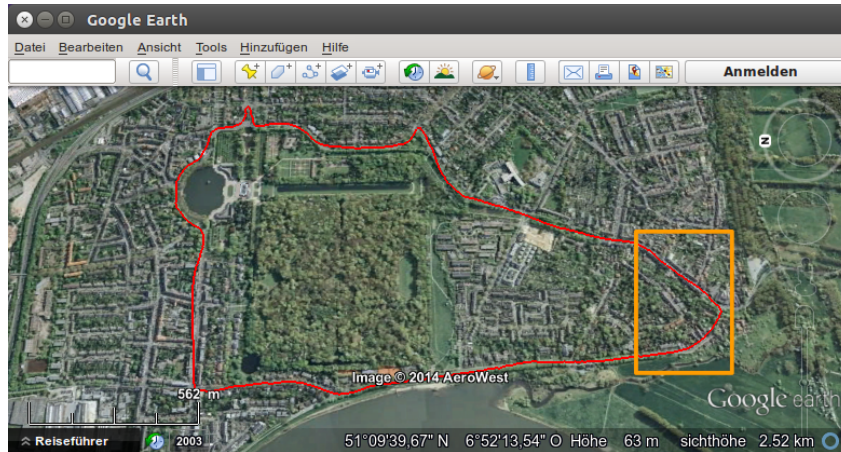


Figure 6.21: Route of the measurement drive shown in Google Earth

streams are centralized managed by the BSC respectively RNC, therefore the overlappings are low. However the upload is divided amongst the terminals and therefore collisions and packet loss arises. However, the uplink has loss of up to 70 %, which is almost unchanged and based on UDP. Additionally the gaps between 150 and 170, 200 and 250 as well as 295 and 315 seconds are obvious. These congestions respectively loss of 100 % happened in the near of the most right part of the red line in Figure 6.21, marked by an orange rectangle. This could be watched in different measurements at this point. Probably there is a handover. As can be seen in Figure 6.22, the data rate is falling shortly before the loss of 100 % occurs. This is detected by SICD and the effects of loss are reduced.

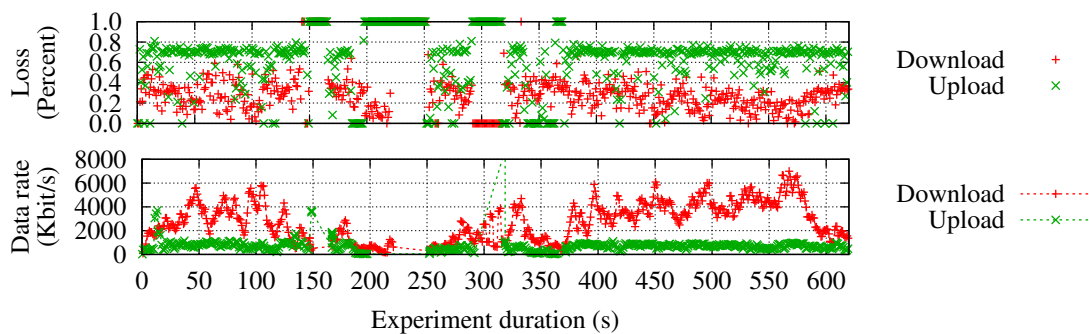


Figure 6.22: Values for the measurement ride on November 4th, 10:55pm without SICD

Figures 6.23 and 6.24 present the delays in transmission as well as the inter-chirp gap times of the same measurement. Every plot of the tripartite figures has a different resolution of the horizontal axis. Additionally the last plot of both figures has a different resolution of the vertical axis. As can be seen, only a few packets have got a delay greater as 5 s. With SICD used, no packet has a

delay greater than 5 s. In the last plot of Figure 6.23 gaps with values close to zero can be seen. This phenomenon only occurs, when chirps are received back-to-back and so congestion occurs. This will be reduced with the measurement ride, which used SICD.

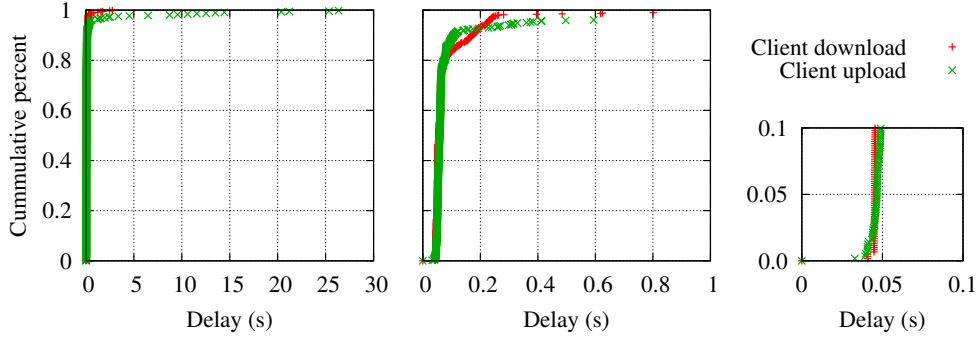


Figure 6.23: CDF for delays of the measurement ride on November 4th, 10:55pm without SICD

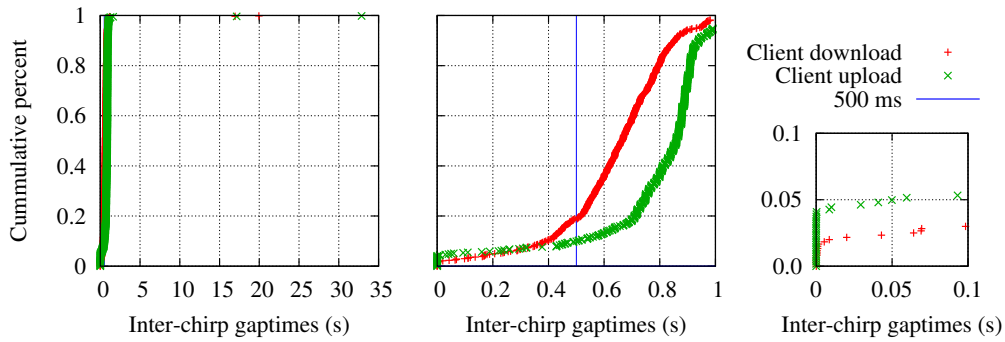


Figure 6.24: CDF for gaps of the measurement ride on November 4th, 10:55pm without SICD

A measurement ride using SICD has been concerned right after the non SICD ride, which increases the significance and comparability of the results. Results of estimated data and loss rate as well as the values of t^{sleep} can be seen in Figure 6.25. Due to different road traffic, the measurement is 30 seconds shorter than the first ride. It is obvious, that the three gaps are missing and loss declines. However, data rate in the clients' upload has got peaks. These peaks are based on congestions, where the complete chirp is congested in the cellular network. This overestimation is possible, when the chirp is congested on a fast link in the network, so that the inter-packet gaps will be reduced. Therefore the packets are theoretically received faster as possible, because for the calculation of the data rate only the receive time stamps are used. Based on the third plot of Figure 6.25, phases with congestions can be detected. Interesting is the small gap at 250th second. This place is the same as the huge gap in Figure 6.22, but now the chirps are not lost due to the detection of congestion. Thereby the terminal concerned stops sending chirps and received a more accurate data rate by the other terminal.

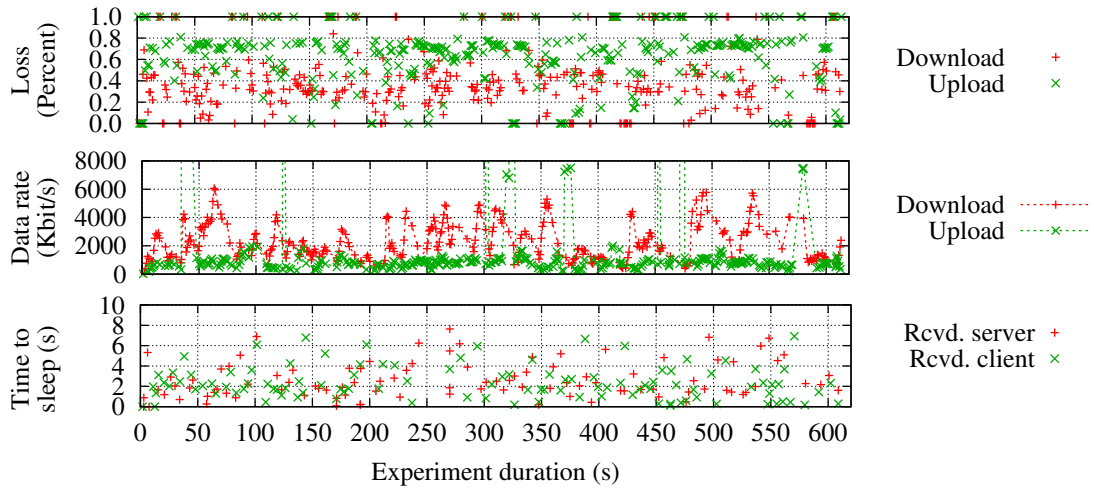


Figure 6.25: Values for the measurement ride on November 4th, 11:05pm, with SICD

Figures 6.26 and 6.27 display CDFs for a measuring at 11:05pm . Arrangement of the plots stays the same as in the previous figures of CDFs. Very important is, that the delays are a lot smaller. Here no packet has a delay greater than 5 s. As can be seen count of gaps with values close to zero are decreased. This can be seen in the second and third plot of Figure 6.27. Especially the download is improved. In conclusion, it can be claimed that the self-induced congestion detector reduced the gaps with values close to zero and therefore congestion.

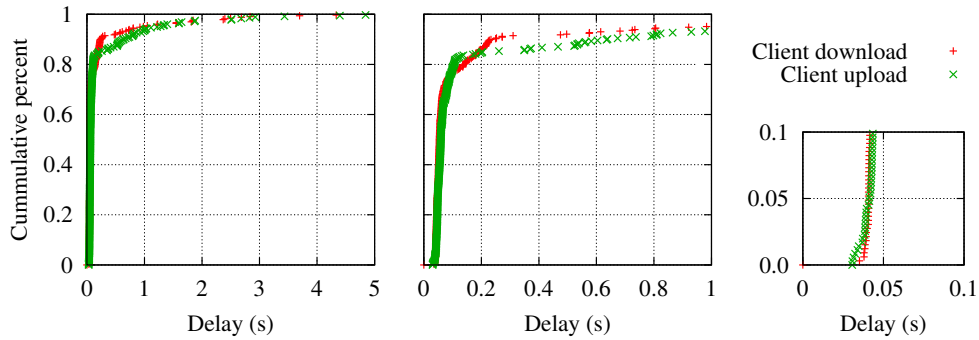


Figure 6.26: CDF for delays of the measurement ride on November 4th, 11:05pm, with SICD

Final Table 6.10 is about statistics on different measurement done in this thesis. Due to lack of space only recent measurements are listed. The columns “chirps rcvd.” as well as “congestions” are divided into the count for the client on the left side and number of the server on the right side. The column of $||t^{gap} < 50ms||$ is divided the vice versa. For example: On November 2th, at 5:00am, an measurement without SICD while moving lasted 297 seconds, whereby the client send 297 chirps and the server 296 chirps. Thereby was one gap in the servers downloads smaller

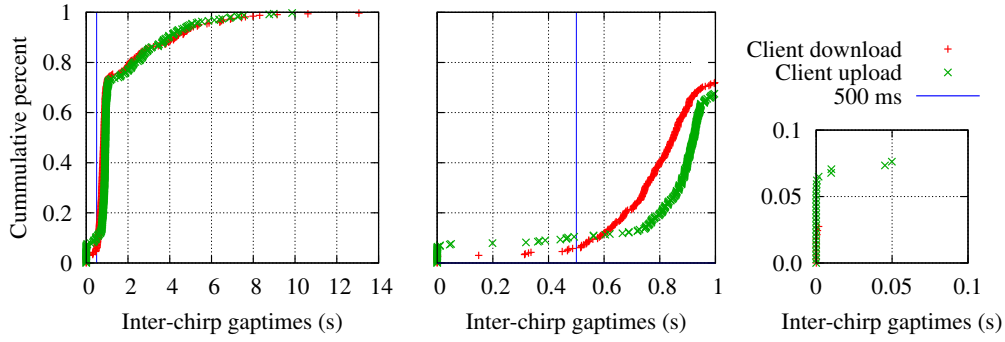


Figure 6.27: CDF for gaps of the measurement ride on November 4th, 11:05pm, with SICD

50 milliseconds. Measurements with stationary client have been done at the authors place or at the data center of the Heinrich-Heine-University. The measurement setup was mentioned in Section 6.2.2. The route of the measurement ride can be seen in Figure 6.21: As can be seen in this statistics, count of $t^{gap} < 50ms$ is not always smaller with use of SICD. This is based on loss phases without SICD, because when no chirp is received, there is no small gap and therefore no congestions; Like in Figure 6.22. Additionally the influences of other users are smaller in the morning than in the evening.

Date	Duration	Rcvd. chirps			$ t^{gap} < 50ms $			SICD?	Moving?	RMF v.
October 31th										
04:10am	600	600	/	600	0	/	1	✗	✗	58842
04:25am	600	600	/	600	0	/	0	✓	✗	58842
04:55am	600	600	/	600	2	/	6	✗	✗	58842
05:15am	600	600	/	600	1	/	5	✓	✗	58842
November 1th										
06:50pm	300	300	/	300	0	/	2	✗	✗	58842
07:00pm	300	300	/	300	0	/	2	✓	✗	58842
08:15pm	300	300	/	300	1	/	1	✗	✗	58842
08:25pm	300	300	/	300	1	/	4	✓	✗	58842
November 2th										
05:00am	297	297	/	296	0	/	1	✗	✓	58842
05:05am	219	279	/	279	2	/	3	✗	✓	58842
05:10am	197	121	/	130	2	/	0	✓	✓	58842
05:20am	247	168	/	154	5	/	4	✓	✓	58842
05:25am	239	169	/	131	5	/	2	✓	✓	58842
05:30am	228	125	/	127	5	/	5	✓	✓	58842
05:35am	203	126	/	119	5	/	4	✓	✓	58842
05:40am	240	164	/	145	10	/	3	✓	✓	58842
05:45am	225	153	/	122	11	/	4	✓	✓	58842
05:50am	252	147	/	140	4	/	2	✓	✓	58842
November 3th										
10:00pm	600	600	/	600	293	/	0	✗	✗	58865
10:25pm	600	590	/	590	0	/	1	✗	✗	58865
November 4th										
10:10am	600	528	/	331	3	/	0	✗	✓	58868
10:15pm	676	655	/	656	25	/	12	✗	✓	58868
10:25pm	675	377	/	368	44	/	26	✓	✓	58868
10:35pm	554	532	/	531	20	/	8	✗	✓	58868
10:45pm	533	402	/	371	24	/	22	✓	✓	58868
10:55pm	647	603	/	566	24	/	5	✗	✓	58868
10:05pm	646	330	/	357	23	/	7	✓	✓	58868
10:15pm	580	563	/	570	11	/	1	✗	✓	58868
10:25pm	578	387	/	387	15	/	15	✓	✓	58868

Table 6.10: Statistics on different measurement in real life

Chapter 7

Summary and Outlook

7.1 Summary

A novel measurement algorithm for logging data rate, drop rate and latency was introduced in this thesis. Therefore various improvements on the RMF were done. In addition models for handling congestion were developed. Inter-chirp congestion can be detected after the first received packet of a chirp or at within a chirp. After congestion is detected, the start time for the next chirp is calculated and send to the other terminal. This terminal will stop sending for the given time, if it does not receives a more recent sleep time. If this time changes due to calculations, intelligent feedback is given. This means, that only necessary values will be send, so congestion on the safeguard channel will be avoided.

Every improvement was tested under laboratory conditions as well as in real situation. For the testbed, evaluations of the most recommended cross-traffic generators were done. Measurement in the testbed have shown shown, that the algorithm can cope with handovers and fallbacks. Unfortunately fast changes in data rates can not be recognized due to measurement with an interval of 1 Hz. Real life measurements using mobile cellular networks have been conducted with stationary clients as well as with moving clients. For both scenarios comparative measurement with and without SICD were conducted. Measurements with fixed position have shown, that there are fewer outliers and the estimated data rate is smoother in contrast to a measurement without SICD used. Measurements while moving have shown, that the amount of chirps send back-to-back can be decreased with the new measurement algorithm. Additionally the high dispersion of the values is decreased and the estimated data rate is smoother, too. Therefore table 6.10 gives a short survey over the amount of measurements done. Summarized it has been shown, that congestion and delays arising from this self-induced inter-chirp congestion can be reduced to a minimum.

7.2 Outlook

The novel measurement algorithm could be enhanced with a database. All measured values can be stored position based data. Thus the algorithm notes places, where self-induced inter-chirp congestion can occur. Therefore, the algorithm will be self-learning and maybe it can cope with changing network characteristics better.

A second improvement is the compatibility with an Android™ smartphone. In an previous bachelors' thesis the RMF was already transferred for Android with the help of the Android™ NDK¹³ (see [Olf13]). The same could be done with the new classes and new measurement algorithm. Therefore dual-SIM and a PPS-signal because time synchronization with cellular networks does not work fine. Additionally a custom Android-system has to be available on the smartphone, so that there are two routes for sending data packets. Currently there are smartphones with Dual-SIMs but from these just one SIM can be active.

The third improvement is a research of the effects of loss for the measurement. As already seen in Section 6.2.2 loss has a negative effect on for example the gap times. Therefore, the measurement algorithm already is able to send safeguard-packets due to high loss on measurements, but no handling for this case was developed due to time.

Last but not least parallel measurement have to be done. Obviously there are many terminals in one mobile cell and every terminal has got different connection properties as well as different hardware. Therefore, the terminals influence each others connections. These should be considered for measurements in and simulations of mobile cellular networks.

¹³See <https://developer.android.com/tools/sdk/ndk/index.html>.

Appendix A

Testbed

This chapter contains results from measurements in the network out of Figure 6.1, thus the names Amy, Bender and Farnsworth are referring to the same network. Section A.1 contains result out of the maximal data rate measurement in the laboratory. Section A.2 deals with packet loss, when traffic is shaped. The third section lists a abbreviated tcpdump of one cross-traffic flow. The last Section A.3 lists scripts, which were used in the laboratory.

A.1 Maximal data rate

Iperf was used, for calculating the maximal available data rate on different parts of the network, which was used for experiments.

Amy acts as client, therefore Iperf needs the IP of the server as well as the parameter for client mode. Additionally, the time and a parameter for duplex measurement are set.

```
1 | praktika@amy:~/krauthoff$ iperf -c 192.168.2.2 -t 60 -d
2 |
3 | Server listening on TCP port 5001
4 | TCP window size: 85.3 KByte (default)
5 |
6 |
7 | Client connecting to 192.168.2.2, TCP port 5001
8 | TCP window size: 65.7 KByte (default)
9 |
10 | [ 5] local 192.168.1.2 port 45942 connected with 192.168.2.2 port 5001
11 | [ 4] local 192.168.1.2 port 5001 connected with 192.168.2.2 port 38330
12 | [ ID] Interval      Transfer    Bandwidth
13 | [ 5]  0.0-60.9 sec  68.0 MBytes  9.37 Mbits/sec
```

```
14 || [ 4] 0.0–62.0 sec 66.9 MBytes 9.05 Mbits/sec
```

Listing A.1: Measuring bandwidth between Amy and Fry: client side

Fry acts as server, therefore Iperf is running as server, no more parameters needed.

```
1 | praktika@fry:~$ iperf -s
2 |
3 | Server listening on TCP port 5001
4 | TCP window size: 85.3 KByte (default)
5 |
6 | [ 4] local 192.168.2.2 port 5001 connected with 192.168.1.2 port 45942
7 |
8 | Client connecting to 192.168.1.2, TCP port 5001
9 | TCP window size: 23.5 KByte (default)
10 |
11 | [ 6] local 192.168.2.2 port 38330 connected with 192.168.1.2 port 5001
12 | [ ID] Interval          Transfer      Bandwidth
13 | [ 6] 0.0–60.7 sec 66.9 MBytes 9.24 Mbits/sec
14 | [ 4] 0.0–62.4 sec 68.0 MBytes 9.15 Mbits/sec
```

Listing A.2: Measuring bandwidth between Amy and Fry: server side

A.2 Packet Loss while using Wondershaper

Wondershaper (see [Hub14b]) is a tool for shaping network traffic, but unfortunately packets will be dropped. Therefore, Farnsworth interfaces were shaped to 2000 kilobyte with the command “sudo wondershaper eth0 2000 2000 && sudo wondershaper eth1 2000 2000”. After this the route between Amy and Fry was measured with Iperf. The Amy’s results are shown in Listing A.3 and Fry’s results are shown in Listing A.4. The measurements shows, that the loss rate rises up to 80%. Amy acts as client, therefore Iperf needs the IP of the server as well as the parameter for client mode. Fry acts as server, therefore Iperf is running as server, no more parameters needed.

Additionally, the time, a parameter for duplex measurement and a parameter for the available data rate are set.

```
1 | praktika@amy:~/krauthoff$ iperf -c 192.168.2.2 -t 60 -d -b 10000K
2 | WARNING: option -b implies udp testing
3 |
4 | Server listening on UDP port 5001
5 | Receiving 1470 byte datagrams
6 | UDP buffer size: 224 KByte (default)
```



```

7 |
8 |
9 | Client connecting to 192.168.2.2, UDP port 5001
10 | Sending 1470 byte datagrams
11 | UDP buffer size: 224 KByte (default)
12 |
13 | [ 5] local 192.168.1.2 port 49400 connected with 192.168.2.2 port 5001
14 | [ 3] local 192.168.1.2 port 5001 connected with 192.168.2.2 port 45416
15 | [ ID] Interval      Transfer      Bandwidth
16 | [ 5] 0.0-60.0 sec  68.6 MBytes  9.59 Mbits/sec
17 | [ 5] Sent 48931 datagrams
18 | [ 5] Server Report:
19 | [ 5] 0.0-60.3 sec  14.0 MBytes  1.95 Mbits/sec   9.925 ms 38929/48929 (80%)
20 | [ 5] 0.0-60.3 sec   3 datagrams received out-of-order
21 | [ 3] 0.0-60.3 sec  14.0 MBytes  1.95 Mbits/sec   9.377 ms 38930/48930 (80%)
22 | [ 3] 0.0-60.3 sec   2 datagrams received out-of-order

```

Listing A.3: Measuring bandwidth between Amy and Fry with shaped router: client side

```

1 | praktika@fry:~/krauthoff$ iperf -s -u
2 |
3 | Server listening on UDP port 5001
4 | Receiving 1470 byte datagrams
5 | UDP buffer size: 224 KByte (default)
6 |
7 | [ 3] local 192.168.2.2 port 5001 connected with 192.168.1.2 port 49400
8 |
9 | Client connecting to 192.168.1.2, UDP port 5001
10 | Sending 1470 byte datagrams
11 | UDP buffer size: 224 KByte (default)
12 |
13 | [ 5] local 192.168.2.2 port 45416 connected with 192.168.1.2 port 5001
14 | [ ID] Interval      Transfer      Bandwidth
15 | [ 5] 0.0-60.0 sec  68.6 MBytes  9.59 Mbits/sec
16 | [ 5] Sent 48931 datagrams
17 | [ 3] 0.0-60.3 sec  14.0 MBytes  1.95 Mbits/sec   9.925 ms 38929/48929 (80%)
18 | [ 3] 0.0-60.3 sec   3 datagrams received out-of-order
19 | [ 5] Server Report:
20 | [ 5] 0.0-60.3 sec  14.0 MBytes  1.95 Mbits/sec   9.377 ms 38930/48930 (80%)
21 | [ 5] 0.0-60.3 sec   2 datagrams received out-of-order

```

Listing A.4: Measuring bandwidth between Amy and Fry with shaped router: server side

A.3 Scripts

The following subsection contains different script for setting the CPU frequency and generating cross traffic with different generators.

A.3.1 CPU Frequency

Some generators (for example Brute) are using the built-in time stamp counter register of the CPU for best temporal accuracy. Therefore, the CPU has to run in performance mode with maximum frequency. So the tool `cpufreq-utils` was used to set the frequency and governor. The script for `cpufreq-utils` is listed in Listing A.5. With the command `etc/init.d/cpufrequtils reload` the script will be used.

```
1 | ENABLE="true"
2 | GOVERNOR="performance"
3 | MAX_SPEED=3100000
4 | MIN_SPEED=3100000
```

Listing A.5: `/etc/default/cpufrequtils`

A.3.2 Cross-Traffic

Listing A.6, A.7 and A.8 contain the scripts for generating cross traffic with Brute, D-ITG and Iperf. Brute was started with the command `sudo taskset -c 3 brute -f traffic.evaluate -d 00:1b:21:9d:50:5f -s e0:69:95:35:e2:c3 -ifin=eth1 -ifout=eth2` the traffic generator can be started, whereby 3 denotes a CPU for pinning the task and `traffic.evaluate` a script for the cross traffic. D-ITG and Iperf were started with the commands listed below.

The parameters `d` and `s` denotes the destination- and source-MAC-address. Everything else is self-explanatory.

```
1 | off msec=5000;
2 | cbr msec=10000; rate=1000; daddr=192.168.6.2; len=200; saddr=192.168.5.2; ↵
   |   ↵sport=9000; dport=9001; ttl=5; tos=8;
3 | cbr msec=10000; rate=1000; daddr=192.168.6.2; len=400; saddr=192.168.5.2; ↵
   |   ↵sport=9000; dport=9001; ttl=5; tos=8;
4 | cbr msec=10000; rate=1000; daddr=192.168.6.2; len=600; saddr=192.168.5.2; ↵
   |   ↵sport=9000; dport=9001; ttl=5; tos=8;
```

```

5 | cbr msec=10000; rate=1000; daddr=192.168.6.2; len=800; saddr=192.168.5.2; ↵
   | ↵sport=9000; dport=9001; ttl=5; tos=8;
6 | cbr msec=10000; rate=1000; daddr=192.168.6.2; len=1000; saddr=192.168.5.2; ↵
   | ↵sport=9000; dport=9001; ttl=5; tos=8;
7 | cbr msec=10000; rate=1000; daddr=192.168.6.2; len=1200; saddr=192.168.5.2; ↵
   | ↵sport=9000; dport=9001; ttl=5; tos=8;
8 | off msec=5000;

```

Listing A.6: Cross-traffic script for brute

In D-ITG -a denotes the destination, -c the frame rate per second, -C the packet length and -t the time in milliseconds.

```

1 | sleep 5
2 | taskset -c 3 ./ITGSend -T UDP -a 192.168.6.2 -c 1000 -C 200 -t 10000
3 | taskset -c 3 ./ITGSend -T UDP -a 192.168.6.2 -c 1000 -C 400 -t 10000
4 | taskset -c 3 ./ITGSend -T UDP -a 192.168.6.2 -c 1000 -C 600 -t 10000
5 | taskset -c 3 ./ITGSend -T UDP -a 192.168.6.2 -c 1000 -C 800 -t 10000
6 | taskset -c 3 ./ITGSend -T UDP -a 192.168.6.2 -c 1000 -C 1000 -t 10000
7 | taskset -c 3 ./ITGSend -T UDP -a 192.168.6.2 -c 1000 -C 1200 -t 10000
8 | taskset -c 3 ./ITGSend -T UDP -a 192.168.6.2 -c 1000 -C 600 -t 10000
9 | sleep 5

```

Listing A.7: Cross-traffic script for D-ITG

In Iperf -c denotes the destination, -t the time in seconds, -b the target bandwidth in bits per second and -u stands for UDP.

```

1 | sleep 5
2 | taskset -c 3 iperf -c 192.168.6.2 -t 10 -b 1600K -u
3 | taskset -c 3 iperf -c 192.168.6.2 -t 10 -b 3200K -u
4 | taskset -c 3 iperf -c 192.168.6.2 -t 10 -b 4800K -u
5 | taskset -c 3 iperf -c 192.168.6.2 -t 10 -b 6400K -u
6 | taskset -c 3 iperf -c 192.168.6.2 -t 10 -b 8000K -u
7 | taskset -c 3 iperf -c 192.168.6.2 -t 10 -b 9600K -u
8 | taskset -c 3 iperf -c 192.168.6.2 -t 10 -b 4800K -u
9 | sleep 5

```

Listing A.8: Cross-traffic script for IPerf

Appendix B

Rate Measurement Framework v. 58894

B.1 Interfaces

Table B.1 contains all interfaces, which the developer has to implemented for creating a new measurement method.

Function	Description
initMeasurement()	Governs the flow of the measurement method, especially the sending queue is filled.
dataFileWriter_measureSender()	Processes data out of the logging-queue.
fastDataCalculator()	A fast data rate calculation with the current packet.
dataFileWriter_measureReceiver()	Processes data out of the receiving-queue.
set_loggingDone()	Introduces the end.
ProcessSafeguardPacket()	Process current safeguard packet.

Table B.1: RMF interfaces for a measurement algorithm

B.2 Parameters

Table B.2 and B.4 contain all parameters for the RMF, Table B.3 is about the settings for the server and client, whereby the parameter `-v` and `-h` will display the versions number respectively the help.

Parameter	Value	Content
-a	0 1	Binding of process / Thread to CPU depending on system specs
-G	0 1	Activate performance scaling governor
-l	0 1 2	LogLevel
-L	0 1	Deactivate swapping of memory
-r	client server	Node Role
-o	folder	Output folder for logfiles
-F	Filename	Filename of experiment without ending
-P	0 1 2	Process scheduling (default round robin fifo)
-Q	Count	Count of max elements per list (memory managment)
-R	Count	max read retry on socket
-W	Count	max write retry on socket
-T	0 1 2 3	Timing Method (0=Hpet, 1=getTimeOfDay, 2=own, 3=autoselect)

Table B.2: Local settings for the RMF

Parameter	Value	Content
-SC	-CC IP	Server / Client Control Channel Address
-SM	-CM IP	Server / Client Measurement Channel Address
-SS	-CS IP	Server / Client Safeguard Channel Address
-Sc	-Cc PORT	Server / Client Control Channel Port
-Sm	-Cm PORT	Server / Client Measurement Channel Port
-Ss	-Cs PORT	Server / Client Safeguard Channel Port
-sc	-cc INTERFACE	Server / Client Control Channel Interface
-sm	-cm INTERFACE	Server / Client Measurement Channel Interface
-ss	-cs INTERFACE	Server / Client Safeguard Channel Interface
-csg	IP	Client Safeguard Channel Gateway

Table B.3: Local settings for the server and client

Parameter	Value	Content
-M	method	Measurement method (basic, assolo, new, mobile, algo)
-m	up, down, both	Measured Link uplink downlink (clients view)
-t	time	Time duration (Sekunden)
-g	on off	Safeguard Channel
-ip4 -ip6		IP Protocoll Version

Table B.4: Global settings for the RMF

Appendix C

Configuration Files

This chapter contains various configuration files for UMTS login, GPSD, NTP and for the Server.

C.1 UMTS Login

Wvdial was used to use of UMTS modems with the Alix Board. Therefore, different providers were used, whereby scripts for Fonice, O2 and T-Mobile can be seen in Listings C.1, C.2 and C.3. What should be noted is that the SIM cards for Fonice and O2 were put in the internal UMTS modem and the T-Mobile-Card was plugged into an Huawei Technologies Co., Ltd. K3765 HSPA stick. Every pin is blacked out with hashtags. Listing C.4 displays additional commands for a general check, a signal check and the command to switch off the modem.

```
1 [Dialer Defaults]
2 Init2 = ATQ0 V1 E1 S0=0 &C1 &D2 +FCLASS=0
3 ISDN = 0
4 Init1 = ATZ
5 Modem = /dev/ttyUSB3
6 Baud = 38400
7
8 [Dialer pin]
9 Init3 = AT+CPIN=####
10
11 [Dialer fonice]
12 Init5 = AT+CGDCONT=1, "IP", "pinternet.interkomm.de", "", 0,0
13 Baud = 38400
14 Username=fonice
15 Password=fonice
16 Dial Command = ATDT
17 Carrier Check = No
18 Phone = *99#
```

```
19 | Stupid Mode = 1
20 | New PPPD = yes
```

Listing C.1: Client: /etc/wvdial.fonic.conf

```
1 | [Dialer Defaults]
2 | Init2 = ATQ0 V1 E1 S0=0 &C1 &D2 +FCLASS=0
3 | ISDN = 0
4 | Init1 = ATZ
5 | Modem = /dev/ttyUSB3
6 | Baud = 38400
7 |
8 | [Dialer pin]
9 | Init3 = AT+CPIN=####
10 |
11 | [Dialer o2]
12 | Carrier Check = No
13 | Init5 = AT+CGDCONT=1, "IP", "internet.debitel", "", 0, 0
14 | Phone = *99#
15 | New PPPD = yes
16 | Username = none
17 | Dial Command = ATDT
18 | Stupid Mode = 1
19 | Password = none
20 | Baud = 38400
```

Listing C.2: Client: /etc/wvdial.o2.conf

```
1 | [Dialer Defaults]
2 | Init2 = ATQ0 V1 E1 S0=0 &C1 &D2 +FCLASS=0
3 | Modem Type = USB Modem
4 | ISDN = 0
5 | Init1 = ATZ
6 | Modem = /dev/ttyUSB7
7 | Baud = 960000
8 |
9 | [Dialer pin]
10 | Init2 = AT+CFUN=1
11 | Init3 = AT+CPIN=####
12 |
13 | [Dialer tmobile]
14 | Carrier Check = no
15 | Init5 = AT+CGDCONT=1, "IP", "internet.t-mobile", ""
16 | Init6 = AT+CSQ
17 | Phone = *99#
18 | New PPPD = yes
19 | Username = tmobile
20 | Dial Command = ATD
21 | Stupid Mode = 1
22 | Password = tm
23 | Dial Attempts = 1
24 | auto dns = 0
```

Listing C.3: Client: /etc/wvdial.tmobile.conf


```
1 | [Dialer signal]
2 | Init1 = AT+CSQ
3 | Init2 = AT+COPS?
4 |
5 | [Dialer check]
6 | Init1 = AT+CPIN?
7 | Init2 = AT+CFUN=?
8 | Init3 = AT+CFUN?
9 |
10 | [Dialer off]
11 | Init1 = AT+CFUN=0
```

Listing C.4: Additional wvdial commands

C.2 **gpsd configurations**

The measurement client had to use GPS while measuring. Therefor `gpsd` was used, so Listing C.5 displays the configuration for the used measurement hardware.

```
1 | # Default settings for gpsd.
2 | # Please do not edit this file directly - use 'dpkg-reconfigure gpsd' to
3 | # change the options.
4 | START_DAEMON="true"
5 | GPSD_OPTIONS="-n -G -F -D 0"
6 | DEVICES="/dev/ttyS4"
7 | USB AUTO="true"
8 | GPSD_SOCKET="/var/run/gpsd.sock"
```

Listing C.5: Client: /etc/default/gpsd

C.3 **NTP configurations**

Correct time stamps are very important for data rate measurements, generally time synchronization is very important for distributed system. Therefor the client used a GPS mouse for being a stratum 1-server. The server used the NTP-server of the data center of the Heinrich-Heine-University Dusseldorf. The servers configuration is listed in Listing C.6 and the clients configuration in Listing C.7. Additionally, the NTP log files should be kept by the operation system, so Listing C.8 describes the edited cron job.

```
1 | # /etc/ntp.conf, configuration for ntpd; see ntp.conf(5) for help
2 | driftfile /var/lib/ntp/ntp.drift
```

```

3
4 # Enable this if you want statistics to be logged.
5 statsdir /var/log/ntpstats/
6 statistics loopstats peerstats clockstats
7 filegen loopstats file loopstats type day enable
8 filegen peerstats file peerstats type day enable
9 filegen clockstats file clockstats type day enable
10
11 # You do need to talk to an NTP server or two (or three) .
12 server time-1.rz.uni-duesseldorf.de minpoll 6 maxpoll 6 iburst
13 server time-2.rz.uni-duesseldorf.de minpoll 6 maxpoll 6 iburst
14
15 # Access control configuration; see /usr/share/doc/ntp-doc/html/accopt.html
16 # for details. The web page might also be helpful:
17 # <http://support.ntp.org/bin/view/Support/AccessRestrictions>
18 #
19 # Note that "restrict" applies to both servers and clients, so a configuration
20 # that might be intended to block requests from certain clients could also end
21 # up blocking replies from your own upstream servers.
22
23 # By default, exchange time with everybody, but don't allow configuration.
24 restrict -4 default kod notrap nomodify nopeer noquery
25 restrict -6 default kod notrap nomodify nopeer noquery
26
27 # Allows the users of the subnet 192.168.2.0 to sync to our server
28 restrict 192.168.2.0
29
30 # Local users may interrogate the ntp server more closely.
31 # Nope, they wont: # restrict 127.0.0.1 # restrict ::1

```

Listing C.6: Server: /etc/ntp.conf

```

1 # /etc/ntp.conf, configuration for ntpd; see ntp.conf(5) for help
2 driftfile /var/lib/ntp/ntp.drift
3
4 # Enable this if you want statistics to be logged.
5 statsdir /var/log/ntpstats/
6 statistics loopstats peerstats clockstats
7 filegen loopstats file loopstats type day enable
8 filegen peerstats file peerstats type day enable
9 filegen clockstats file clockstats type day enable
10
11 # You do need to talk to an NTP server or two (or three) .
12 server 127.127.28.0 true minpoll 4 maxpoll 4
13 fudge 127.127.28.0 time1 0.510 refid GPS
14 server 127.127.28.1 minpoll 4 prefer true
15 fudge 127.127.28.1 refid PPS
16
17 # Access control configuration; see /usr/share/doc/ntp-doc/html/accopt.html
18 # for details. The web page might also be helpful:
19 # <http://support.ntp.org/bin/view/Support/AccessRestrictions>
20 #
21 # Note that "restrict" applies to both servers and clients, so a configuration
22 # that might be intended to block requests from certain clients could also end

```

```

23 # up blocking replies from your own upstream servers.
24
25 # By default, exchange time with everybody, but don't allow configuration.
26 restrict -4 default kod notrap nomodify nopeer noquery
27 restrict -6 default kod notrap nomodify nopeer noquery
28
29 # Allows the users of the subnet 192.168.2.0 to sync to our server
30 restrict 192.168.2.0
31
32 # Local users may interrogate the ntp server more closely.
33 restrict 127.0.0.1
34 restrict ::1

```

Listing C.7: Client: /etc/ntp.conf

```

1 #!/bin/sh
2
3 # The default Debian ntp.conf enables logging of various statistics to the
4 # /var/log/ntpstats directory. The daemon automatically changes to a new
5 # dated set of files at midnight, so all we need to do is delete old
6 # ones, and compress the ones we're keeping so disk usage is controlled.
7
8 statsdir=$(cat /etc/ntp.conf | grep -v '^#' | sed -n 's/statsdir \([^ ]*\) ↵
↵ ]*\)/\1/p')
9
10 if [ -n "$statsdir" ] && [ -d "$statsdir" ]; then
11     # only keep a week's depth of these
12     # nope, don't delete it
13     # find "$statsdir" -type f -mtime +7 -exec rm {} \;
14
15     # compress whatever is left to save space
16     cd "$statsdir"
17     ls loopstats.???????? peerstats.???????? > /dev/null 2>&1
18     if [ $? -eq 0 ]; then
19         # Note that gzip won't compress the file names that
20         # are hard links to the live/current files, so this
21         # compresses yesterday and previous, leaving the live
22         # log alone. We suppress the warnings gzip issues
23         # about not compressing the linked file.
24         gzip --best --quiet loopstats.???????? peerstats.????????
25         return=$?
26         case $return in
27             2)
28                 exit 0          # squash all warnings
29                 ;;
30             *)
31                 exit $return    # but let real errors through
32                 ;;
33         esac
34     fi
35 fi

```

Listing C.8: Server: /etc/cron.daily/ntp

C.4 ARP-Reply's and Source-Based-Routing

In this thesis the server had two IP-addresses in the same subnet. Due to purposes of security and safety the first three octets of the IP address are denoted as $x.y.z$. Both addresses are in same the subnet with the first 27 bit set. Now every interface has to answer for the right incoming packets. Therefore, the ARP Reply's on wrong devices had to be disabled first, so the kernel parameters can be modified. All commands are listed in Listing C.9.

```
1 net.ipv4.conf.eth0.arp_filter=1
2 net.ipv4.conf.eth0.arp_ignore=1
3 net.ipv4.conf.eth0.arp_announce=2
4
5 net.ipv4.conf.eth1.arp_filter=1
6 net.ipv4.conf.eth1.arp_ignore=1
7 net.ipv4.conf.eth1.arp_announce=2
```

Listing C.9: Server: /etc/sysctl.conf

Additionally, both IP addresses are set as static parameter. The configuration of both devices can be seen in Listing C.10.

```
1 auto lo
2 iface lo inet loopback
3
4 # The primary network interface
5 auto eth0
6 iface eth0 inet static
7     address x.y.z.228
8     netmask 255.255.255.224
9     network x.y.z.224
10    broadcast x.y.z.255
11    gateway x.y.z.225
12
13 auto eth1
14 iface eth1 inet static
15     address x.y.z.229
16     netmask 255.255.255.224
17     network x.y.z.224
18     broadcast x.y.z.255
19     gateway x.y.z.225
```

Listing C.10: Server: /etc/network/interfaces

Now source-based-routing has to be enabled. First there must be a IP table for every interface. The content of */etc/iproute2/rt_tables* is listed in Listing C.11. Afterwards the commands for routing are set in the *rc.local*-file, see Listing C.12.

```
1 || 1    eth0
2 || 2    eth1
```

Listing C.11: Server: /etc/iproute2/rt_tables

```
1 || #add source based routing
2 || ip route add default via x.y.z.225 dev eth0 table eth0
3 || ip route add x.y.z.224/27 dev eth0 src x.y.z.228 table eth0
4 || ip rule add from x.y.z.228 table eth0
5 || ip rule add oif eth0 table eth0
6 ||
7 || ip route add default via x.y.z.225 dev eth1 table eth1
8 || ip route add x.y.z.224/27 dev eth1 src x.y.z.229 table eth1
9 || ip rule add from x.y.z.229 table eth1
10 || ip rule add oif eth1 table eth1
```

Listing C.12: Server: /etc/rc.local

After a reboot, the command *ip route show* should display the output in Listing C.13, whereby both tables are filled like the output in Listing C.14 for eth0 and Listing C.15 for eth1.

```
1 || Define the routes for these two tables:
2 || default via x.y.z.225 dev eth1
3 || x.y.z.224/27 dev eth0 proto kernel scope link src x.y.z.228
4 || x.y.z.224/27 dev eth1 proto kernel scope link src x.y.z.229
```

Listing C.13: Server: ip route show

Define the rules for when to use the new routing tables:

```
1 || default via x.y.z.225 dev eth0
2 || x.y.z.224/27 dev eth0 scope link src x.y.z.228
```

Listing C.14: Server: ip route show table eth0

```
1 || default via x.y.z.225 dev eth1
2 || x.y.z.224/27 dev eth1 scope link src x.y.z.229
```

Listing C.15: Server: ip route show table eth1

Appendix D

Miscellaneous

The sections are about gpsd, gpspipe and ntpq data.

D.1 GPSD Data

Listing D.1 displays a snapshot of the data, which were shared by the GPS-daemon.

```
1 { "class": "TPV", "tag": "GGA", "device": "/dev/ttyS0", "mode": 3, ↵
  ↵ "time": "2014-07-15T06:58:15.000Z", "ept": 0.005, "lat": 51.186648333, ↵
  ↵ "lon": 6.797103333, "alt": 63.500, "epx": 3.257, "epy": 4.615, ↵
  ↵ "epv": 11.868, "track": 128.4000, "speed": 0.000, "eps": 16.65 }
2 { "class": "SKY", "tag": "GSV", "device": "/dev/ttyS0", "xdop": 0.87, ↵
  ↵ "ydop": 1.23, "vdop": 2.06, "tdop": 1.27, "hdop": 1.51, "gdop": 2.85, ↵
  ↵ "pdop": 2.56, "satellites": [ ↵
  ↵ { "PRN": 5, "el": 8, "az": 189, "ss": 26, "used": true }, ↵
  ↵ { "PRN": 15, "el": 71, "az": 271, "ss": 28, "used": true }, ↵
  ↵ { "PRN": 17, "el": 22, "az": 115, "ss": 21, "used": true }, ↵
  ↵ { "PRN": 18, "el": 27, "az": 305, "ss": 40, "used": true }, ↵
  ↵ { "PRN": 24, "el": 34, "az": 267, "ss": 35, "used": true }, ↵
  ↵ { "PRN": 8, "el": 3, "az": 75, "ss": 0, "used": false }, ↵
  ↵ { "PRN": 22, "el": 3, "az": 334, "ss": 0, "used": false }, ↵
  ↵ { "PRN": 26, "el": 62, "az": 136, "ss": 0, "used": false }, ↵
  ↵ { "PRN": 28, "el": 46, "az": 58, "ss": 0, "used": false }, ↵
  ↵ { "PRN": 30, "el": 13, "az": 77, "ss": 0, "used": false } ] }
3 { "class": "TPV", "tag": "GGA", "device": "/dev/ttyS0", "mode": 3, ↵
  ↵ "time": "2014-07-15T06:58:16.000Z", "ept": 0.005, "lat": 51.186648333, ↵
  ↵ "lon": 6.797103333, "alt": 63.500, "epx": 3.257, "epy": 4.615, ↵
  ↵ "epv": 11.868, "track": 128.4000, "speed": 0.000, "eps": 16.65 }
```

Listing D.1: Data from the gps receiver

D.2 Gpspipe Data

Listing D.2 displays an extract of logged data by the tool `gpspipe` (see [Mil14a]). The data was fetched on start up of `gpspipe`, therefore the first two lines contain information about the version of `gpspipe`, as well as the path, where the data is fetched. The third line contains parameters, which could be set. The last lines, which are denoted by SKY and TPV contains the same information like above.

```

1 Tue Jul 15 08:01:32 2014 :{"class":"VERSION", "release":"3.4", "rev":"3.4", ↵
  ↵"proto_major":3, "proto_minor":6}
2 Tue Jul 15 08:01:32 2014 :{"class":"DEVICES", "devices":[{"class":"DEVICE", ↵
  ↵"path":"/dev/ttyS0", "activated":"2014-07-15T06:01:32.769Z", ↵
  ↵"flags":1, "driver":"Generic NMEA", "native":0, "bps":19200, ↵
  ↵"parity":"N", "stop$
3 Tue Jul 15 08:01:32 2014 :{"class":"WATCH", "enable":true, "json":true, ↵
  ↵"nmea":false, "raw":0, "scaled":false, "timing":false}
4 Tue Jul 15 08:01:33 2014 :{"class":"TPV", "tag":"GGA", ↵
  ↵"device":"/dev/ttyS0", "mode":3, "time":"2014-07-15T06:01:33.000Z", ↵
  ↵"ept":0.005, "lat":51.186468333, "lon":6.797085000, "alt":63.400, ↵
  ↵"epx":2.442, "epy":2.288, "epv$
5 Tue Jul 15 08:01:33 2014 :{"class":"SKY", "tag":"GSV", ↵
  ↵"device":"/dev/ttyS0", "xdop":0.65, "ydop":0.61, "vdop":1.12, ↵
  ↵"tdop":0.56, "hdop":0.89, "gdop":1.54, "pdop":1.43, ↵
  ↵"satellites":[{"PRN":5, "el":32, "az":196, "ss":29, "

```

Listing D.2: First lines out of the log from `gpspipe`

D.3 NTPQ Data

Listing D.3 shows the NTP data for a GPS-mouse and the two timeserver of the data center of the Heinrich-Heine-University Dusseldorf with the help of the command `ntpq -p`. Additionally, the fallback-server of Ubuntu is listed.

	remote	refid	st	t	when	poll	reach	delay	offset	jitter
1	=====									
2	=====									
3	+time-1.rz.uni-d	.PPS.	1	u	5	64	37	0.727	-0.233	0.098
4	-time-2.rz.uni-d	.PPS.	1	u	4	64	17	0.793	-0.327	0.263
5	+SHM(0)	.GPS.	0	l	5	16	377	0.000	120.017	16.650
6	*SHM(1)	.PPS.	0	l	4	16	377	0.000	0.000	0.015
7	juniperberry.ca	.INIT.	16	u	-	64	0	0.000	0.000	0.000

Listing D.3: `ntpq -p`

Bibliography

- [3GP09] 3GPP. 3GPP TS 23.009: Handover procedures. http://www.3gpp.org/ftp/Specs/archive/23_series/23.009/23009-820.zip, September 2009. Last checked: 15th July 2014.
- [3GP14a] 3GPP. 3GPP. <http://www.3gpp.org/>, July 2014. Last checked: 17th July 2014.
- [3GP14b] 3GPP. GPRS & EDGE. <http://www.3gpp.org/technologies/keywords-acronyms/102-gprs-edge>, July 2014. Last checked: 20th July 2014.
- [3GP14c] 3GPP. HSPA. <http://www.3gpp.org/technologies/keywords-acronyms/99-hspa>, July 2014. Last checked: 20th July 2014.
- [3GP14d] 3GPP. LTE. <http://www.3gpp.org/technologies/keywords-acronyms/98-lte>, July 2014. Last checked: 20th July 2014.
- [3GP14e] 3GPP. LTE-Avanced. <http://www.3gpp.org/technologies/keywords-acronyms/97-lte-advanced>, July 2014. Last checked: 20th July 2014.
- [3GP14f] 3GPP. UMTS. <http://www.3gpp.org/>, July 2014. Last checked: 20th July 2014.
- [ABBH⁺14] Cerion Armour-Brown, Jeremy Borntraeger, Christian and Fitzhardinge, Tom Hughes, Petar Jovanovic, Dejan Jevtic, Florian Krohm, Carl Love, Maynard Johnson, Paul Mackerras, Dirk Mueller, Nicholas Nethercote, Julian Seward, Bart Van Assche, Robert Walsh, Philippe Waroquiers, Josef Weidendorfer, Frederic Gobry, Daniel Berlin, Nick Clifton, Michael Matz, and Simon Haus-

- mann. Valgrind Home. <http://valgrind.org/>, June 2014. Last checked: 10th June 2014.
- [AG] Brian Adamson and Sean Gallavan. The multi-generator (mgen) toolset.
- [APV04] Stefano Avallone, Antonio Pescapé, and Giorgio Ventre. Analysis and experimentation of Internet Traffic Generator. 2004. Proc. of New2an, pages 70–75.
- [BD14] Dominik Brodowski and Mattia Dongili. Index of pub linux utils kernel cpufreq. <https://www.kernel.org/pub/linux/utils/kernel/cpufreq/cpufreq-info.html>, September 2014. Last checked: 15th September 2014.
- [BDP10] Alessio Botta, Alberto Dainotti, and Antonio Pescapé. Do you trust your software-based traffic generator? 2010. Communications Magazine, IEEE, pages 158 - 165.
- [BDPGP12] Nicola Bonelli, Andrea Di Pietro, Stefano Giordano, and Gregorio Procissi. Flexible high performance traffic generation on commodity multi-core platforms. In Antonio Pescapé, Luca Salgarelli, and Dimitropoulos Xenofontas, editors, *Traffic Monitoring and Analysis*, pages 157–179. Springer, 2012.
- [BGPS05] Nicola Bonelli, Stefano Giordano, Gregorio Procissi, and Raffaello Secchi. BRUTE: A High Performance and Extensible Traffic Generator. 2005. Proc. of SPECTS, pages 839–845.
- [Bon14] Nicola Bonelli. brute - High performance network traffic generator - Google Project Hosting. <https://code.google.com/p/brute/>, August 2014. Last checked: 19th Aug 2014.
- [Bou14] Peter Bourgon. Who needs boost? a simple pthreads wrapper. <http://peter.bourgon.org/blog/2010/10/27/who-needs-boost-a-simple-pthreads-wrapper.html>, July 2014. Last checked: 22th July 2014.
- [CSNO14] China Satellite Navigation Office. BeiDou navigation satellite system. <http://en.beidou.gov.cn/>, August 2014. Last checked: 18th August 2014.
- [Dal93] P Daly. Navstar GPS and GLONASS: global satellite navigation systems. 1993. IET, pages 349–357.

- [DC14] Voyage Design and Consultants. Voyage linux { x86 Embedded Linux = Green computing }. <http://linux.voyage.hk/>, September 2014. Last checked: 19th Sep 2014.
- [Eri14] Marius Eriksen. trickle. <http://monkey.org/~marius/pages/?page=trickle>, August 2014. Last checked: 05th August 2014.
- [ESA14a] European Space Agency. What is galileo? http://www.esa.int/Our_Activities/Navigation/The_future_-_Galileo/What_is_Galileo, August 2014. Last checked: 18th August 2014.
- [ESA14b] European Space Agency. What is glonass? http://www.esa.int/About_Us/ESA_Permanent_Mission_in_Russia/GLONASS, August 2014. Last checked: 18th August 2014.
- [ETS14] ETSI. ETSI TS 123 272. http://www.etsi.org/deliver/etsi_ts/123200_123299/123272/11.09.00_60/ts_123272v110900p.pdf, July 2014. Last checked: 15th July 2014.
- [Fon14] Jose Fonseca. Gprof2dot - wiki. <https://code.google.com/p/jrfonseca/wiki/Gprof2Dot>, August 2014. Last checked: 20th August 2014.
- [Fre14] GNU Free. Wvdial - ArchWiki. <https://wiki.archlinux.org/index.php/Wvdial>, July 2014. Last checked: 1th July 2014.
- [FS14] Jay Fenlason and Richard Stallman. Gnu gprof - table of contents.html. https://www.cs.utah.edu/dept/old/texinfo/as/gprof_toc.html, August 2014. Last checked: 20th August 2014.
- [fVudI14] Bundesministerium für Verkehr und digitale Infrastruktur. Bundesministerium für verkehr und digitale infrastruktur. <http://www.bmvi.de/SharedDocs/DE/Artikel/UI/galileo-das-europaeische-satellitennavigationssystem.html>, August 2014. Last checked: 21th August 2014.
- [GKMK14] Norbert Goebel, Markus Koegel, Martin Mauve, and Graffi Kalman. Trace-based Simulation of C2X-Communication using Cellular Networks. March

2014. In: Wireless On-demand Network Systems and Services (WONS), 2014 11th Annual Conference on. Pages 108–115.
- [Goe10] Norbert Goebel. Trace-basierte Simulation von Mobilfunkcharakteristiken für die Fahrzeug-zu-Fahrzeug Kommunikation. Master’s thesis, Heinrich-Heine-Universität Düsseldorf, August 2010.
- [Gra13] Kalmann Graffi. Mobilkommunikation - Technische Grundlagen. Lecture at Heinrich-Heine-Universität Düsseldorf, in the summer-term 2013, April 2013. Last checked: 20th July 2014.
- [GRT09] Emanuele Goldoni, Giuseppe Rossi, and Alberto Torelli. Assolo, a New Method for Available Bandwidth Estimation. May 2009. Internet Monitoring and Protection, 2009. ICIMP ’09. Fourth International Conference, page 130–136.
- [Hem14] Stefan Hemminger. Mainpage of tc-netem. <http://man7.org/linux/man-pages/man8/tc-netem.8.html>, September 2014. Last checked: 28th September 2014.
- [HS03] Ningning Hu and Peter Steenkiste. Evaluation and Characterization of Available Bandwidth Probing Techniques. April 2003. In: Selected Areas in Communications, IEEE Journal on, page 879–894.
- [Hub14a] Bert Hubert. Mainpage of tc. <http://lartc.org/manpages/tc.txt>, September 2014. Last checked: 23th September 2014.
- [Hub14b] Bert Hubert. Wonder Shaper. <http://lartc.org/wondershaper/>, July 2014. Last checked: 01th July 2014.
- [Inc14] Google Incorporation. gperftools - Fast, multi-threaded malloc() and nifty performance analysis tools. <https://code.google.com/p/gperftools/>, August 2014. Last checked: 20th August 2014.
- [Ind14] Indian Space Research Organisation. Indian Space Research Organisation. <http://www.isro.org/index.aspx>, August 2014. Last checked: 18th August 2014.
- [itPI14] Software in the Public Interest. Debian – Das universelle Betriebssystem. <https://www.debian.org/index.de.html>, September 2014. Last checked: 19th

Sep 2014.

- [J⁺96] Rick Jones et al. Netperf: a network performance benchmark. 1996.
- [JD02] Manish Jain and Constantinos Dovrolis. Pathload: A Measurement Tool for End-to-End Available Bandwidth. 2002. In Proceedings of Passive and Active Measurements (PAM) Workshop, pages 14–25.
- [Kel14a] Walt Kelly. NTP Debugging Techniques. <http://doc.ntp.org/4.2.0/debug.html?advanced=on>, June 2014. Last checked: 23th June 2014.
- [Kel14b] Walt Kelly. ntpq - standard NTP query program. <http://doc.ntp.org/4.2.4/ntpq.html>, June 2014. Last checked: 09th June 2014.
- [KGL] C Krasic, A Goel, and K Li. The mxtraf network traffic generator.
- [KM07] Petr Kovar and Karol Molnar. Precise Time Synchronization over GPRS and EDGE. 2007. Personal Wireless Communication (Boston: Springer), pages 277-284.
- [Kul14] Mirko Kulpa. Iperf - The TCP/UDP Bandwidth Measurement Tool. <https://iperf.fr/>, August 2014. Last checked: 09th August 2014.
- [Lam14] Bies Lammert. Gps time synchronization tutorial. <http://www.lammertbies.nl/comm/info/GPS-time.html>, July 2014. Last checked: 23th July 2014.
- [Lan13] Christian Lange. Untersuchung der Auswirkung paralleler Datenratenmessungen in einer Mobilfunkzelle. Master’s thesis, Heinrich-Heine-Universität Düsseldorf, June 2013.
- [LB90] Kevin Lai and Mary Baker. Measuring Bandwidth. March 1990. In: INFO-COM ’99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Pages 235–245.
- [LBA04] Tanja Lang, Philip Branch, and Grenville Armitage. A synthetic traffic model for Quake3. 2004. Proceedings of the 2004 ACM SIGCHI International Conference on Advances in computer entertainment technology, pages 233–238.

- [LDS06] Lao Li, Constantine Dovrolis, and Medy Sanadidi. The Probe Gap Model can Underestimate the Available Bandwidth of Multihop Paths. October 2006. ACM SIGCOMM Computer Communication Review Homepage archive Volume 36 Issue 5, page 29–34.
- [Ler12] Reinhard Lerch. *Elektrische Messtechnik - Analoge, digitale und computergestützte Verfahren*. Springer Vieweg, 6. edition, 2012. Page 475.
- [Lin13] Wilfried Linder. Einführung in die Geoinformatik, April 2013. Lecture at Heinrich-Heine-Universität Düsseldorf, in the winter-term 2012/2013.
- [LSP02] Juha Laine, Sampo Saaristo, and Rui Prior. Rude/crude. 2002.
- [LTWW93] Will E Leland, Murad S Taqqu, Walter Willinger, and Daniel V Wilson. On the self-similar nature of ethernet traffic. 1993. ACM SIGCOMM Computer Communication Review, pages 183–193.
- [MD90] J.C. Mogul and S.E. Deering. Path MTU discovery. RFC 1191 (Draft Standard), November 1990.
- [Mil85] D.L. Mills. Network Time Protocol (NTP). RFC 958, September 1985. Obsoleted by RFCs 1059, 1119, 1305.
- [Mil88] D.L. Mills. Network Time Protocol (version 1) specification and implementation. RFC 1059, July 1988. Obsoleted by RFCs 1119, 1305.
- [Mil89] D.L. Mills. Network Time Protocol (version 2) specification and implementation. RFC 1119 (Standard), September 1989. Obsoleted by RFC 1305.
- [Mil92] D. Mills. Network Time Protocol (Version 3) Specification, Implementation and Analysis. RFC 1305 (Draft Standard), March 1992. Obsoleted by RFC 5905.
- [Mil13] David Miller. ethtool (8) - Linux manual page. <http://linux.die.net/man/8/ethtool>, July 2013. Last checked: 7th August 2014.
- [Mil14a] Gary Miller. gpspipe. <http://catb.org/gpsd/gpspipe.html>, June 2014. Last checked: 08th June 2014.

- [Mil14b] Gary Miller. Pulse-Per-Second (PPS) Signal Interfacing. <http://www.eecis.udel.edu/~mills/ntp/html/pps.html>, July 2014. Last checked: 21th July 2014.
- [MN98] Makoto Matsumoto and Takuji Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. 1998. ACM Transactions on Modeling and Computer Simulation (TOMACS), pages 3–30.
- [Nav14] Naval Research Laboratory. Multi-Generator (MGEN) Networks and Communication Systems Branch. <http://www.nrl.navy.mil/itd/ncs/products/mgen>, September 2014. Last checked: 12th Sep 2014.
- [NCOfSBPNT14] National Coordination Office for Space-Based Positioning, Navigation, and Timing. GPS.gov: GPS-Overview. <http://www.gps.gov/systems/gps/>, July 2014. Last checked: 20th July 2014.
- [Olf13] Malte Olfen. Ende-zu-Ende-Messungen von Mobilfunkparametern mit Smartphones. Bachelor’s thesis, June 2013.
- [QSS14] QSS. Quasi-Zenith Satellite System. <http://www.qzs.jp/en/>, August 2014. Last checked: 18th August 2014.
- [RCR⁺00] Vinay Ribeiro, M. Coates, R. Riedi, S. Sarvotham, and R. Baraniuk. Multifractal Cross Traffic Estimation. September 2000. Proc. of ITC specialist seminar on IP traffic Measurement.
- [RKM⁺14] Eric Raymond, Chris Kuethe, Gary Miller, Remco Treffcorn, Derrick Brashar, and Russ Nelson. gpsd. <http://catb.org/gpsd/gpsd.html>, June 2014. Last checked: 08th June 2014.
- [RN14] Rusty Russel and Michael Neuling. Mainpage of iptables. <http://ipset.netfilter.org/iptables.man.html>, September 2014. Last checked: 29th September 2014.
- [Rou14] Gaël Roualland. ifstat - Report InterFace STATistics. <http://manpages.ubuntu.com/manpages/dapper/man1/ifstat.1.html>, September 2014. Last checked: 15th September 2014.
- [RRBLC03] Vinay Ribeiro, Rudolf Riedi, Richard Baranuik, and Jiri Navratil Les Cottrel. patchchirp: Efficient Available Bandwidth Estimation for Network Paths. Au-


- gust 2003. In: Passive and Active Monitoring Workshop (PAM 2003).
- [Sch03] Jochen Schiller. *Mobilkommunikation*. Addison-Wesley/Pearson Studium, 2. edition, 2003.
- [Wil12] Sebastian Wilken. Verfahren zur Datenratenmessung in Mobilfunknetzen. Master's thesis, Heinrich-Heine-Universität Düsseldorf, June 2012.
- [WK14] Thomas Williams and Coling Kelley. gnuplot homepage. <http://www.gnuplot.info/>, July 2014. Last checked: 26th July 2014.
- [Yoc14] Yocto Project, A Linux Foundation Collaborative Project. EGLIBC: News. <http://www.eglibc.org/news>, August 2014. Last checked: 12th Aug 2014.
- [Zan] Sebastian Zander. Udpngen-udp kernel traffic generator for linux.
- [ZKA05] Sebastian Zander, David Kennedy, and Grenville Armitage. Kute– A High Performance Kernel-based UDP Traffic Engine. 2005.

Declaration of Authorship

I hereby declare that the thesis submitted is my own unaided work. All direct or indirect sources used are acknowledged as reference. I have not used other than the declared sources, resources, and that I have explicitly marked all material which has been quoted either literally or by content from other used sources. This paper was not previously presented to another examination board and has not been published

Dusseldorf, 7. November 2014

Tobias Krauthoff



stick cover

with CD/DVD here

This disk contains:

- *pdf*-version of the present thesis
- \LaTeX - and graphic-source codes of the present thesis together with all scripts used
- source codes as part of the thesis' measurement methods
- data sets used for evaluations
- websites of the internet sources used