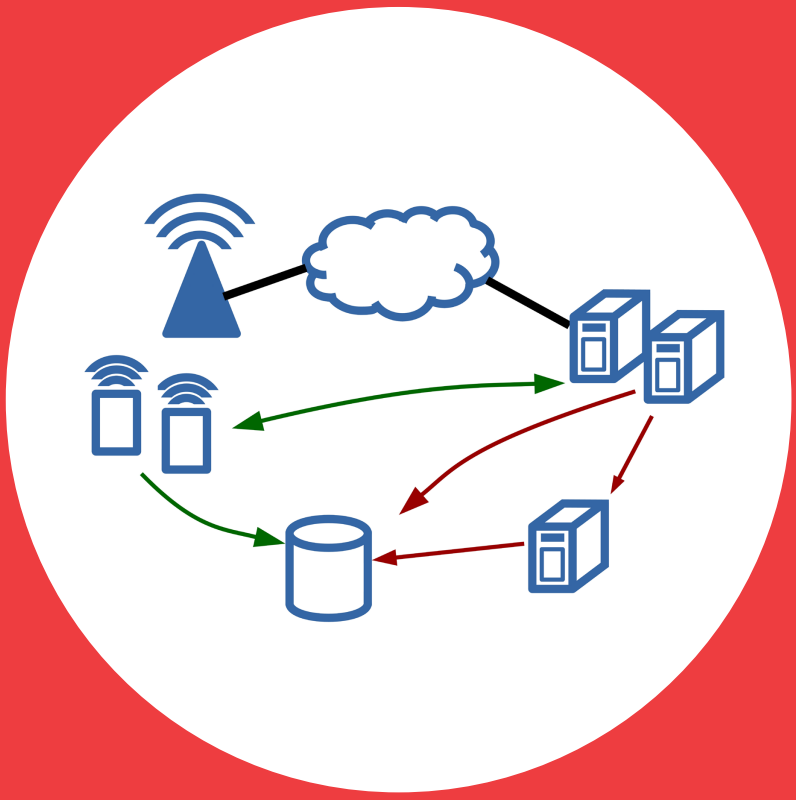
Department of Communications and Networking

Mobile network delay characteristics and interactions with the transport layer

Lennart Schulte



Aalto University

DOCTORAL DISSERTATIONS

Aalto University publication series
DOCTORAL DISSERTATIONS 244/2017

Mobile network delay characteristics and interactions with the transport layer

Lennart Schulte

A doctoral dissertation completed for the degree of Doctor of Science (Technology) to be defended, with the permission of the Aalto University School of Electrical Engineering, at a public examination held at the lecture hall T2 of the school on 19 January 2018 at 12.

Aalto University
School of Electrical Engineering
Department of Communications and Networking

Supervising professor
Professor Jukka Manner, Aalto University, Finland

Thesis advisor
Professor Jukka Manner, Aalto University, Finland

Preliminary examiners
Doctor Tapio Frantti, University of Oulu, Finland
Professor Michael Welzl, University of Oslo, Norway

Opponent
Professor Jussi Kangasharju, University of Helsinki, Finland

Aalto University publication series
DOCTORAL DISSERTATIONS 244/2017

ISBN 978-952-60-7761-1 (printed)
ISBN 978-952-60-7762-8 (pdf)
ISSN-L 1799-4934
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)
http://urn.fi/URN:ISBN:978-952-60-7762-8

Unigrafia Oy
Helsinki 2017

Finland

441 697
Printed matter

Aalto University

Abstract

Aalto University, P.O. Box 11000, FI-00076 Aalto www.aalto.fi

Author
Lennart Schulte

Name of the doctoral dissertation
Mobile network delay characteristics and interactions with the transport layer

Publisher School of Electrical Engineering

Unit Department of Communications and Networking

Series Aalto University publication series DOCTORAL DISSERTATIONS 244/2017

Field of research Networking technology

Manuscript submitted 11 April 2017 **Date of the defence** 19 January 2018

Permission to publish granted (date) 31 October 2017 **Language** English

☐ **Monograph** ☒ **Article dissertation** ☐ **Essay dissertation**

Abstract

Mobile networks have become an integral part of every day life, and many users rely on its presence and performance. In the past years user num bers and traffic volume has skyrocketed, as adoption of the technology continues to increase.

The popularity of internet based services in mobile networks and the increase in user numbers makes the available resources precious and ever more important to use them: a user's satisfaction is inversely proportional to the time it takes for the requested content to load.

While the mobile network sets an upper bound on the transmission rate, it is up to the delivering end point to make use of these resources, which for many applications is in the hands of the Transmission Control Protocol (TCP). As TCP has to make the best use of the resources in every situation, it is vital to understand the interactions between the protocol and the mobile network in order to achieve best performance.

This work is a measurement based study on the performance of TCP in Finnish 3rd generation (3G) and 4th generation (4G) mobile networks. First, it is investigated how the round-trip time (RTT) behaves throughout end-to-end connections traversing mobile networks, on long term as well as on short term in the form of delay spikes. Second, the proper way for TCP to deal with these delay spikes is examined. Lastly, the interactions of TCP with the mobile network is investigated in real-life situations and causes for sub-optimal performance is extracted. The thesis concludes with a discussion on what is necessary to improve TCP performance in mobile networks, and the changes coming with future networks and algorithms.

Keywords mobile networks, TCP, performance measurement, interactions

ISBN (printed) 978-952-60-7761-1	**ISBN (pdf)** 978-952-60-7762-8	
ISSN-L 1799-4934	**ISSN (printed)** 1799-4934	**ISSN (pdf)** 1799-4942
Location of publisher Helsinki	**Location of printing** Helsinki	**Year** 2017
Pages 140	**urn** http://urn.fi/URN:ISBN:978-952-60-7762-8	

Preface

My first thanks goes to Professor Jukka Manner for giving me the opportunity to start my doctoral studies. He supervised my work from beginning to completion. His continuous support, practical approach, and constructive feedback brought me to where I am today.

Next, thanks to my diploma thesis supervisor and friend Alexander Zimmermann. His advice and persistence has taught me how to approach problems methodically, without which I would not be able to pursue a PhD degree. Additionally, he commended me to professors in Finland, which subsequently led me to move here.

Additionally, special thanks shall go to Sebastian Sonntag. The first few months in Finland were quite overwhelming. Sebastian was always there to support me and answer all my questions, which made the beginning so much easier!

Then, there were two post-docs who tried to increase my scientific and paper writing skills: Mikko Särelä, and Nuutti Varis. I would like to thank both of them for their effort, discussions, and comments.

I also could not have done all the work without the other co-authors: Eren Boz, Gautam Moktan, and Lars Eggert. Discussions with and comments from each one have been highly valuable to me. The papers would not have been published with them. Additionally, thanks to Árpád Drozdy for his insight into mobile networks and providing feedback for papers and this thesis.

From a technical perspective, a big thank you goes to the whole Netradar team: Arttu Tervo, Jukka-Pekka Virtanen, Antti Jaakkola and Tommi Tuura. Without their continuous effort this work would not have been possible.

Last, but definitely not least, my thanks goes to my wife Pia. Without her accompanying me to Finland, none of this would have happened. She

always supported me and believed in me even when I did not. Additionally, she always listened to me even when the topic was probably not as close to her heart as it was for me. On a related note, I should also thank my son Joonas for providing distraction from work. Great job!

Espoo, Saturday 25th November, 2017,

Lennart Schulte

Contents

Preface 1

Contents 3

List of Publications 5

Author's Contribution 7

Acronyms 9

1. Introduction 11

1.1 Motivation 12

1.2 Research assumptions, questions, and methodology 13

1.3 Contribution 14

1.4 Structure of the thesis 15

2. Background 17

2.1 Mobile networks 17

2.2 Measurement setup 19

2.2.1 Netradar 19

2.2.2 Mobile network performance 20

2.3 Transmission Control Protocol 22

2.3.1 Core functionality 22

2.3.2 Extensions 23

2.3.3 Analysis 24

2.4 Related work 25

2.5 Summary 27

3. Delay characteristics 29

3.1 Delay during data transfer 29

3.1.1 RTT composition 29

3.1.2 Methodology 30
3.1.3 Results 30
3.2 Interruptions 32
3.2.1 Definition 32
3.2.2 Methodology 32
3.2.3 Results 33
3.3 Packet reordering 35
3.3.1 Definition 35
3.3.2 Methodology 36
3.3.3 Results 38
3.4 Summary 41

4. Protocol characteristics **43**
4.1 Reordering reaction 43
4.1.1 TCP with reordering 43
4.1.2 Algorithm 44
4.1.3 Methodology 45
4.1.4 Results 47
4.2 TCP efficiency 48
4.2.1 TCP and network queueing 48
4.2.2 Algorithm 48
4.2.3 Results 49
4.3 Summary 53

5. Conclusion **55**
5.1 Research questions 55
5.2 Discussion 56
5.3 Future networks and algorithms 58

References **61**

Publications **69**

List of Publications

This thesis consists of an overview and of the following publications which are referred to in the text by their Roman numerals.

I Sebastian Sonntag, Lennart Schulte, Jukka Manner. Mobile network measurements – It's not all about signal strength. In *IEEE Wireless Communications and Networking Conference (WCNC)*, Shanghai, China, pp. 4624–4629, DOI: 10.1109/WCNC.2013.6555324, April 2013.

II Sebastian Sonntag, Jukka Manner, Lennart Schulte. Netradar – Measuring the wireless world. In *9th International Workshop on Wireless Network Measurements (WiNMee)*, Tsukuba Science City, Japan, pp. 29–34, May 2013.

III Lennart Schulte, Gautam Moktan, Mikko Särelä, Jukka Manner. Towards understanding interruption and reordering in HSPA/HSPA+ networks. In *Seventh International Workshop on Selected Topics in Mobile and Wireless Computing (STWiMob)*, Larnaca, Cyprus, pp. 51–57, DOI: 10.1109/WiMOB.2014.6962149, October 2014.

IV Sebastian Sonntag, Lennart Schulte, Jukka Manner. No links left behind. *Elsevier Computer Communications (ComCom)*, Volume 111, 1 October 2017, Pages 97-104, DOI: 10.1016/j.comcom.2017.07.007, July 2017.

V Lennart Schulte, Alexander Zimmermann, Puneeth Nanjundaswamy, Lars Eggert, Jukka Manner. I'll be a bit late – Packet Reordering in Mobile Networks. Accepted for publication in *KICS Journal of Communications and Networks (JCN)*, March 2017.

VI Lennart Schulte, Eren Boz, Nuutti Varis, Jukka Manner. On detecting TCP path saturation in LTE networks. *Wiley International Journal of Communication Systems (IJCS)*, DOI: 10.1002/dac.3334, April 2017.

Author's Contribution

Publication I: "Mobile network measurements – It's not all about signal strength"

This work analyzes the parameters which affect connection quality with special respect to location, movement, and time. Schulte's contribution was centered around the design and analysis of time based diversity in performance. The paper was joint work.

Publication II: "Netradar – Measuring the wireless world"

The focus of this paper was the analysis of parameters from mobile phones, especially throughput quality from the perspective of mobile phones. Schulte's contribution was the design and implementation as part of the Netradar team. The paper was written together.

Publication III: "Towards understanding interruption and reordering in HSPA/HSPA+ networks"

This paper describes connection interruption and packet reordering in mobile networks and finds impacting context factors. Schulte's contribution was the design, implementation and analysis of the measurements. The paper was written together.

Publication IV: "No links left behind"

The article measures Multipath TCP and single TCP connections when aggregating multiple mobile networks. Schulte analyzed the latency. Measurement setup and paper writing was joint work.

Publication V: "I'll be a bit late – Packet Reordering in Mobile Networks"

In this article, the packet reordering characteristic in mobile networks is studied in depth, and a reordering reaction algorithm is studied with respect to mobile networks and latency. Schulte performed implementation and analysis of the measurement study, as well as the methodology and analysis of the algorithm measurements. Implementation of the algorithm, setup for analyzing the algorithm, methodology of the measurement study, the algorithm itself, and paper writing was joint work.

Publication VI: "On detecting TCP path saturation in LTE networks"

The article studies the efficiency of TCP in mobile networks. Schulte's contribution was the methodology, implementation and analysis. The paper was written as a team.

List of Abbreviations

2G	2nd Generation.
3G	3rd Generation.
4G	4th Generation.
5G	5th Generation.
ABC	Appropriate Byte Counting.
ACK	Acknowledgment.
AQM	Active Queue Management.
ARQ	Automatic Repeat Request.
CDN	Content Distribution Network.
CWND	Congestion Window.
DSACK	Duplicate SACK.
DUPACK	Duplicate ACK.
DUPTHRESH	Duplicate ACK Threshold.
ECN	Explicit Congestion Notification.
EDGE	Enhanced Data Rates for GSM Evolution.
EWMA	Exponentially Weighted Moving Average.
F-RTO	Forward RTO-Recovery.
GPRS	General Packet Radio Service.
HARQ	Hybrid ARQ.
HSPA	High Speed Packet Access.
HTB	Hierarchical Token Bucket.

IP	Internet Protocol.
IW	Initial Window.
LTE	Long-Term Evolution.
LTE-A	LTE-Advanced.
MIMO	Multiple Input Multiple Output.
MTU	Maximum Transmission Unit.
OS	Operation System.
PCAP	Packet Capture (format).
PEP	Performance Enhancing Proxy.
RFC	Request for Comments.
RTO	Retransmission Timeout.
RTT	Round-Trip Time.
RWND	Receiver Window.
SACK	Selective ACK.
SIM	Subscriber Identity Module.
SWND	Sender Window.
TCP	Transmission Control Protocol.
TCP-aNCR	Adaptive Non-Congestion Robustness for TCP.
UDP	User Datagram Protocol.
UMTS	Universal Mobile Telecommunications System.
WLAN	Wireless Local Area Network.

1. Introduction

While computer networking has been established for at least 60 years, the foundation for the internet was created only 35 years ago. At that time the ARPANET received additional funding, which broadened the access to this network. With the introduction of the internet protocol suite (TCP/IP) in 1982 and interconnectivity with the NSFNET project in 1986, the internet was born.

Since then, its popularity has spread from researchers to companies, and finally to private households. The internet has become a natural resource for information, communication, and entertainment for everyone.

For a long time, internet access was provided with wired technologies. Only in the 1990s this changed with the first data transfers through mobile networks, and wireless local area networks (WLANs) in private households. The rise of mobile networks with internet access has had a big impacted on society. Now, information and entertainment are available to you wherever you go.

Mobile networks have become an integral part of everyday life with many users relying on its presence and performance. This also shows in the usage numbers. According to Cisco [32], in the past 10 years the global mobile data traffic has increased 4000-fold. The report looks in depth at the year 2015: in this year mobile data traffic grew by 74 %. The year 2015 was also the first year where 4th generation (4G) mobile network traffic exceeded 3rd generation (3G) traffic. In total, 563 million new mobile devices and subscriptions were ordered. Cisco also predicts, that in 2020 the global mobile data traffic will have increased further by a factor of 8.

1.1 Motivation

The popularity of mobile devices, especially smartphones and tablets, has increased dramatically over the last few years. With it, the technology involved in mobile networks, wireless link to the user devices as well as the core network, has grown to be more complex to support the vast amount of users and the needs of different applications.

One goal of building the networks and services is the satisfaction of the mobile user. While mobile networks have evolved rapidly, all components of the data transfer have to collaborate in order to achieve the best outcome.

As such, there are more components as just the mobile network to consider: while bigger service providers make use of content distribution networks (CDNs) in the mobile core network, many services the user wants to access reside within the internet. All data must pass through the internet and then the mobile network, forming a heterogeneous data path transfer.

A substantial portion of the accessed data in terms of data volume is served as bulk transfer, for example mobile apps or video files. Mobile apps have to be downloaded before they can be installed and used, and for every update the installation process is performed again. Video files can either be downloaded directly or viewed via streaming. Streaming services such as YouTube still have a portion of bulk transfer in the beginning when the video buffers until enough data has been received. This can take seconds or even tens of seconds. This time is critical, as the user is more likely to abandon watching if the waiting times are longer, which might consequently reduce the revenue for the content provider [51].

The underlying protocol for these data transfers in the internet is often the Transmission Control Protocol (TCP). It assures that all data reaches its destination and provides algorithms to prevent the network from being overloaded. Hence, while the mobile network provides a natural upper bound on the transmission rate of the connection, it is TCP that controls the sending rate of the connection. It must be able to cope with the changing conditions in mobile networks.

Therefore, in order to reach the most efficient data transfer, TCP and the mobile network have to cooperate. Only then is a critical bottleneck in the data transfer mitigated, and the data transfer takes the least amount of time.

1.2 Research assumptions, questions, and methodology

This work studies the interactions between TCP and the mobile network to answer the question if these are optimal, and if not, what would be needed to improve the situation in the future.

The analysis is a measurement study based on real-world connections. The real-world connections are part of the data set collected by Netradar, a crowd sourced measurement platform for mobile devices. This work uses the full bulk download throughput as a basis. As such, the long term effects are measured. The research assumption is that mobile networks require reliable packet delivery service which can cope with the changing conditions imposed by the network.

The first part of this thesis consists of investigating the round-trip time (RTT) in mobile networks and how it behaves. Measuring how the network behaves, and forwards data is a preliminary step that shows what end-to-end measurements have to cope with. The RTT analysis starts with an overview based on single User Datagram Protocol (UDP) pings to get a general understanding on the distribution of the RTT. The ping measurements are part of the Netradar data set. These statistics are followed by an in-depth view into the RTT characteristics during data transfer. First, the overall effect of queueing in the network is measured, and events are studied that increase the RTT suddenly, like connection interruption and packet reordering.

Subsequently, the interactions between mobile network and TCP are studied. The in-depth characterization of the RTT is used as a basis for observing how the network properties affect the end-to-end performance. The first part looks at the TCP reaction to packet reordering in mobile networks. The second part investigates how TCP performs in the mobile network environment.

To summarize, there are three questions asked in this work:

- How does the RTT behave in mobile networks?
- How should TCP react to events causing delay spikes?
- How well does TCP perform in mobile networks, and what has to change to improve the behavior?

1.3 Contribution

The contribution is split up into six separate publications, each investigating a part of this thesis.

First, the thesis author was part of the team that build the Netradar measurement platform. Two papers study the behavior of the gathered parameters: Publication I and Publication II. Publication I investigates the correlation between measured signal strength and TCP throughput. Publication II gives a broader overview on the measurement methodology in Netradar. Furthermore, it shows the impact of hour of the day, device, and movement speed on the performance.

The Netradar platform is extended to incorporate mechanisms for a deeper study of TCP, especially the recording of packet traces and additional TCP related parameters on the Netradar servers.

Second, the RTT behavior in mobile networks is analyzed in Publication III, Publication IV and Publication V. A part of Publication IV shows the RTT during bulk TCP transfers. These measurements are used to describe the increase of the RTT due to queueing in the network. Publication III consists of 3G measurements regarding connection interruption and reordering. The paper investigates the impacting factors from the context and surroundings. Publication V provides a deeper view on the reordering characteristic and its behavior in 3G and 4G networks. For both of these studies about reordering, the reordering detection algorithm is specified in an IETF Internet-Draft [87]. The 4G measurements regarding connection interruption are added in this thesis. The general statistics on the ping RTT in mobile networks are also created solely for this thesis.

Third, the interactions between mobile network and TCP are investigated in Publication V and Publication VI. The second part of Publication V is the investigation of a new reordering reaction algorithm regarding changes in the network sending rate and the added delay. The algorithm is specified in an IETF Internet-Draft [88]. Publication VI then shows the efficiency of TCP in mobile networks and investigates the events that lead to performance degradation. The algorithm is heavily based on the RTT progression throughout the connection.

1.4 Structure of the thesis

The rest of the thesis is structured as follows. Each of the chapters includes material or knowledge from one or more of the publications.

Chapter 2 starts by introducing mobile networks and their development over time. It highlights the importance of the technology for the population. Subsequently, the Netradar measurement platform is introduced and described as a mean for investigation throughout the thesis. Based on the data, a general overview of delay distributions is presented for mobile networks.

Chapter 3 investigates the delay characteristics more closely, especially with respect to events that increase the delay. First, the increase of delay due to data transfer and queueing in the network is shown. Second, the occurrence and frequency of connection interruptions and packet reordering is analyzed, as being two events that produce delay spikes.

Chapter 4 presents the behavior of TCP in the mobile network context. First, the analysis for a reordering reaction algorithm is presented. Second, TCP efficiency in mobile networks is analyzed and culprit events are found.

Chapter 5 summarizes and concludes the thesis. Additionally, the work presented in the previous chapters is put into context of upcoming mobile technologies and TCP modifications are discussed which could help to increase the performance.

2. Background

This chapter starts by describing mobile networks, their evolution and importance. Then, the measurement setup is introduced and an overview of mobile network performance is given, based on the crowd-sourced measurement platform Netradar. Lastly, the methodology for analyzing TCP [70] connections on top of Netradar is shown.

This chapter is based on Publication I and Publication II.

2.1 Mobile networks

The work in this thesis is focused on the measurement of end-to-end path characteristics of data transfer in a heterogeneous data path where one part is a mobile network. Hence, a desription of the mobile network is presented to get an understanding of the involved technologies.

While the first releases of 2nd generation (2G) networks [1] were circuit switched and optimized for voice telephony, rudimentary data transfer was already possible. The first packet switched Internet Protocol (IP) [69] based internet became available to customers with the deployment of General Packet Radio Service (GPRS), or "2.5G" which was standardized in 1998. Shortly after, the introduction of enhanced data rates for GSM evolution (EDGE) to 2G networks increased the available data transfer rate to several 100 kbit/s. In all newer releases data transfer and internet access was improved dramatically.

The first 3G release was universal mobile telecommunications system (UMTS) [5], which was standardized in 2000 with deployment starting in 2004. The European 2G project created much attention so that 3G was object to global standardization. In terms of bit rate, the initial UMTS release had a peak rate of 384 kbit/s. With the introduction of High Speed Packet Access (HSPA) [2] it was gradually improved to reach

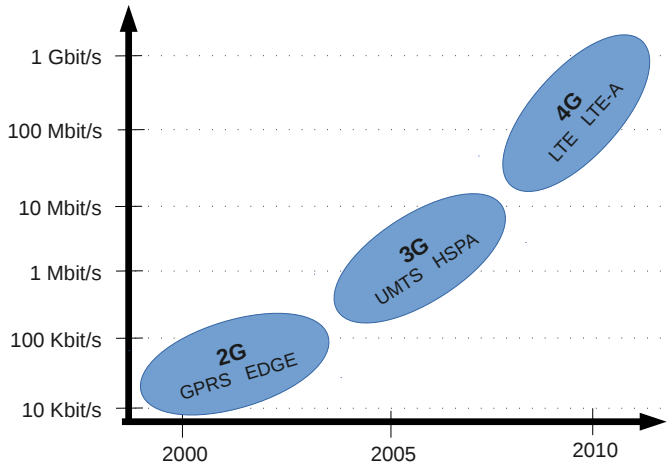


Figure 2.1. Mobile network data rate.

tens of Mbit/s.

Even higher packet rates are achieved with currently deployed 4G technology. The Long-Term Evolution (LTE) [3] technology promises data rates of 100 Mbit/s, and up to 1 Gbit/s with LTE-Advanced [4]. The main difference to former 3G networks is a simpler and flat core network, which allows for faster connection set-up time and handovers. A rough overview of the technologies and their theoretically achievable data rates is presented in Figure 2.1.

In order to achieve higher data rates, new technologies were introduced in HSPA which have significant impact on the analysis performed in this work: hybrid automatic repeat request (HARQ), and link allocation in the base station [33]. HARQ performs retransmissions on the wireless link much quicker than automatic repeat request (ARQ), which reduces waiting times for corrupted frames. In modern networks, HARQ is tried first for a fast resending of missing frames. Only if it fails ARQ is invoked. The link allocation was moved from the core network to the base stations, which makes the process for allocation much faster: it is now able to adapt to the fast-changing conditions of the link to reallocate resources between users every few ms. Giving more resources to users where the channel conditions are better in the short run improves overall throughput for everyone in the long run. Due to the significant differences to previous technologies and the focus towards modern mobile networks, in this work I analyze only HSPA and LTE networks.

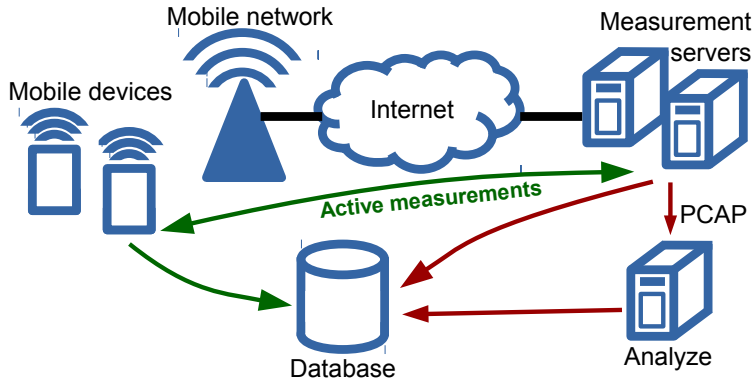


Figure 2.2. Netradar measurement platform.

2.2 Measurement setup

This section explains the measurement platform that is used to gather the data needed for a large-scale measurement study. Then, general performance characteristics for Finnish mobile networks are presented.

2.2.1 Netradar

In order to infer relevant statistics about data transfer through mobile networks, it is important to have as much data as possible. To this extent, the Netradar measurement platform [64] was used as a basis. Netradar consists of servers in the internet and mobile phone apps for many smartphone platforms, i.e. iPhone, Android, Windows Phone, and many Nokia smartphones. The setup is shown in Figure 2.2.

The smartphone app collects context information and performs active measurements. The collection of context information is designed to extract as much information about the surroundings and smartphone status as possible, i.e. any information which might correlate and influence the actively performed measurements. Context information includes the mobile device, i.e. vendor, model, and OS version; the mobile network, i.e. which operator, technology, and basestation; as well as battery status, location, and application layer signal strength.

The active measurements are performed against one of the Netradar measurement servers. The server to be used is assigned to the client for each measurement so that the amount of measurements against each server can be regulated. All active measurements are performed consecutively to avoid interference between the measurements. The client first determines the end-to-end RTT to the server by sending small UDP [71]

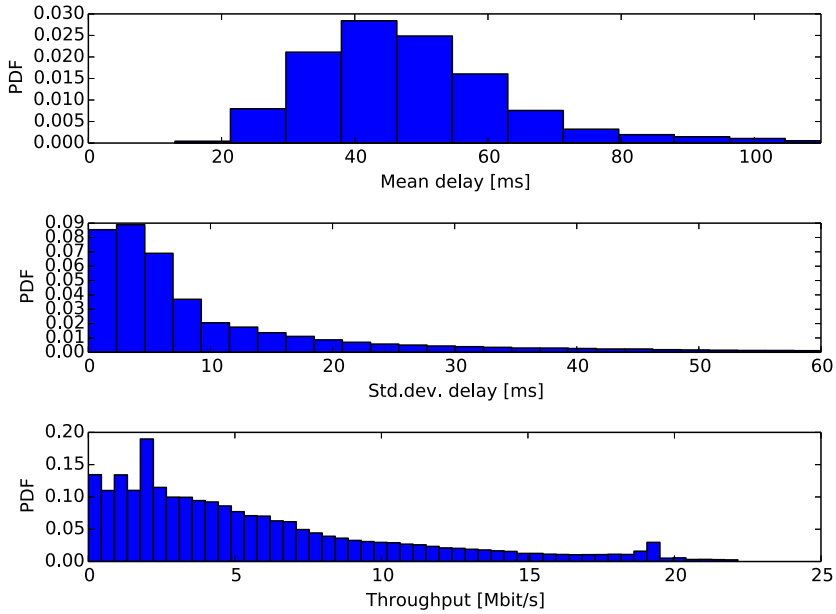


Figure 2.3. Distributions for RTT and throughput in 3G HSPA networks.

messages. The server replies as soon as the message arrives. For each of the 10 sent pings the difference between send time and return time is calculated. After that, first, a download throughput test is performed and then an upload throughput test. Both are 10 s TCP bulk transfers of random data to determine the maximum sending rate. The sender uses TCP Cubic for all bulk transfer measurements.

2.2.2 Mobile network performance

Overviews of the networks main characteristics is given in Figure 2.3 for 3G HSPA networks, and in Figure 2.4 for 4G networks. These show the user perceived performance of data transfer through mobile networks. Both figures show the mean RTT, the standard deviation for RTT, and the average throughput for the Netradar data of 2015. Only successful measurements are taken into account. In total, there were 190 000 data points for 3G and 575 000 for 4G.

The 3G networks in Figure 2.3 show a delay in the range 20 ms to 100 ms with a peak around 40 ms. The standard deviation is generally below 10 ms, but can be much higher. The throughput is often below 10 Mbit/s, and naturally capped at 21 Mbit/s.

The 4G networks in Figure 2.4 have a mean delay between 10 ms to 50 ms, a standard deviation mostly below 5 ms and throughput of up to

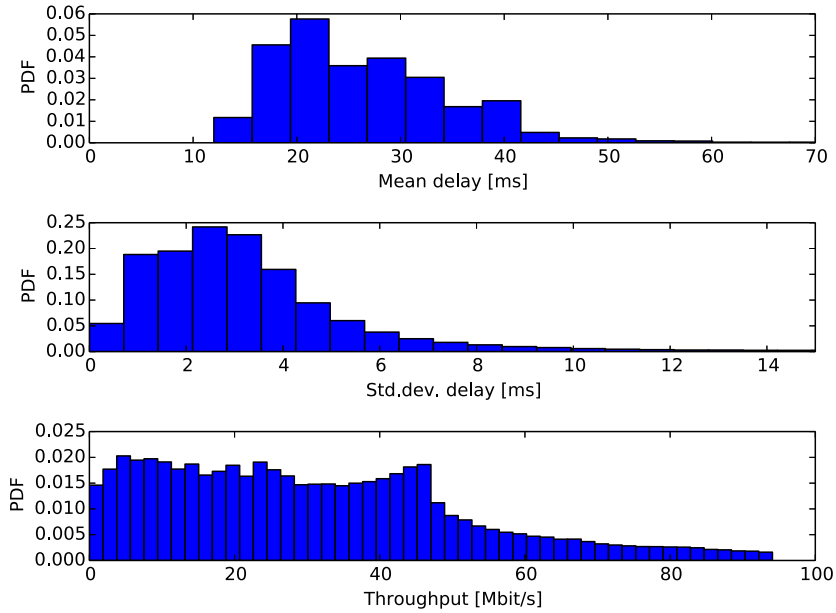


Figure 2.4. Distributions for RTT and throughput in 4G networks.

90 Mbit/s. For the throughput, we see cap often applied by operators in Finnish mobile networks of 50 Mbit/s. Hence, in contrast to 3G networks, the 4G networks have about half the delay with half the standard deviation, and a throughput which is much higher.

Looking more at the performance of 4G networks, Figure 2.5 shows the quarterly average throughput for three operator networks over the course from 2014 to 2016. While operator 3 stays fairly even, the throughput in the other two operator networks declines over the time. This is most visible for operator 2, where the highest point of 42 Mbit/s is reached in early 2014. In 2016 the same operator only reaches about 28 Mbit/s on average. Hence, the performance has dropped by over 30 %. This can be explained by the increase in user numbers, either new users entirely or users switching from 3G to 4G. Since the available bandwidth is limited, a higher number of users have to compete for the same resources, and the throughput for individual users is decreased.

Overall, we see that the performance in 4G networks has much improved over 3G networks, even over HSPA. But the increasing user numbers decrease the performance for individual users. This highlights the importance to make the most out of the offered resources, in order to achieve the best user experience and satisfaction.

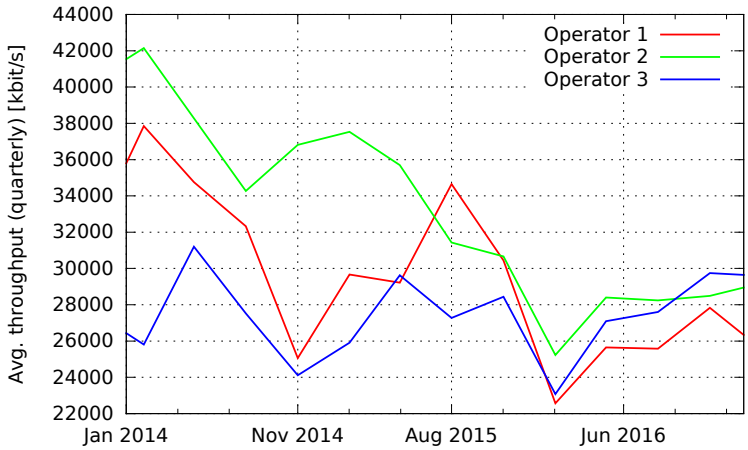


Figure 2.5. Throughput of 4G networks over the last 2 years.

2.3 Transmission Control Protocol

The mobile network offers resources to the user, and is therefore a natural limit on the transmission rate. It is in the best interest of the user as well as the network provider to make the best use out of these resources during a data transfer: the user receives the requested data faster which increases satisfaction; the network provider is interested in the efficient resource usage as unused resources might have been better allocated to different users.

How to use the resources is normally controlled at the endpoints, and more specifically at the transport layer. One example for this is TCP [70], which is the underlying protocol for file transfers, web content, and video downloads.

2.3.1 Core functionality

TCP [70] enables the transmission of data between two endpoints in both directions. The data is transmitted reliably, which means that TCP ensures that all sent data arrives correctly at the receiver. To achieve this, all data is cumulatively acknowledged: the receiver sends back an acknowledgment (ACK) that tells up to which point the data has been received correctly. When data is missing, the receiver sends the same cumulative ACK as before, called a duplicate acknowledgment (DUPACK). When three DUPACKs are received, i.e. the duplicate acknowledgement threshold (DUPTHRESH) is reached, the sender retransmits the missing data [9] and enters recovery until the missing data has been acknowl-

edged. Additionally, this event is seen as an indication for congestion in the network, and hence, the sending rate is reduced.

The congestion control algorithm is also responsible for increasing the sending rate throughout the connection, which is done while not in recovery. The sending rate is controlled by the congestion window (CWND), which tells how many segments can be sent into the network at any given time. First, in the beginning of the connection the CWND is set to the initial window (IW) of 10 segments [31]. Then, the slow start algorithm [9] increases the CWND by one for every ACK, which results in an exponential gain. After the first recovery until the end of the connection, congestion avoidance is used instead of slow start, which increases the CWND much slower.

The standardized congestion control algorithm is Reno [9] which increases the CWND by one every RTT, and halves the CWND when entering recovery. A very popular one, and the default in the Linux operation system (OS) is Cubic congestion control [75] which increases with a cubic function, hence the name, and decreases the CWND by 30 % in recovery. There are many other approaches as well [60, 28, 30].

Congestion or severe path changes are also assumed, when the sender sees no response from the receiver for the duration of the retransmission timeout (RTO) [67]. In this case, the CWND is set to 1 and slow start is used again to probe the network. In addition, all outstanding data is seen as lost and retransmitted.

TCP also implements flow control, and makes sure that neither of the endpoints is overloaded. It limits the CWND of the connection according to the announced receiver window (RWND) and sender window (SWND), which tells how much buffer is available.

Hence, while the network offers resources, it is in the hands of the used transport layer protocol, in this case TCP, to make use of them. Underutilized network resources are undesirable for network providers, content providers, as well as users. TCP therefore has to be well-designed in order to achieve satisfactory performance in the challenging and highly adaptable environment of modern mobile networks.

2.3.2 Extensions

There are many extensions to improve TCP performance in a variety of situations [35], especially when recovering from losses. The most important ones for this work are explained here.

The selective acknowledgment (SACK) option [61] makes it possible for the receiver to tell the sender about packets, that have been received out-of-order, meaning that data is missing but newer data has arrived which cannot be cumulatively acknowledged. Based on this option, the recovery is much faster when facing multiple packet losses [21].

The TCP Timestamps option [24] adds a timestamp to each packet, which is echoed by the receiver in the ACK. With this option it is possible to calculate the RTT for retransmitted packets, as otherwise it is unclear whether the original transmission or the retransmission has triggered the ACK. This is called the retransmission ambiguity [50].

Nowadays, it is very common for the receiver to only send an ACK for every two incoming packets [25]. This substantially reduces processing overhead, as it reduces the number of packets to be processed. When facing dubious situations, for example missing segments, an ACK is still sent immediately.

The calculation of the RTO is based on the RTT. Hence, the RTO might be spuriously triggered if no response (an ACK) is incoming due to network outage or delay variation. In such cases, there are two extensions which can detect if it was spurious and revert the reaction of necessary. The Eifel algorithm [59, 58] detects a spurious RTO with the help of timestamps. Each retransmission is accompanied by a new timestamp. Hence, if the first arriving ACK after entering the loss recovery is accompanied by the timestamp of the original data, then the RTO was triggered spuriously, and the changes can be reverted. Alternatively, the Forward RTO-Recovery (F-RTO) algorithm [79] provides a similar functionality without using timestamps. It transmits new data on the first ACK after the RTO and determines spuriousness based on the following ACK.

2.3.3 Analysis

In this work, the analysis is performed on the basis of TCP connections. Hence, it is important to find as much information about the protocol as possible.

The decision making of the protocol is performed at the sender side, while the receiver only acknowledges data. Hence, gathering information on the sender side is more important than on the receiver side for the purpose of this thesis.

The measurement servers run the Linux [82] OS Ubuntu [27]. There are two methods of obtaining information about the TCP state on the

servers which are used: packet traces, and asking the OS for the current state. The current state can be asked from the OS via the socket option `TCP_INFO`. It returns the status of many TCP parameters such as the CWND. But this information has to be asked every time the status should be recorded, which means that important data might be missed when it is not asked often enough. Still, asking too often infers a needlessly huge amount of data that has to be stored and processed. Hence, this information is collected every 50 ms by the throughput test application running on the server and sent to a database. The 50 ms is a trade-off which gathers information about every 1 to 2 RTTs in mobile networks. This is enough for longer running metrics.

The second, and more important data that is used are packet traces, which are much more accurate since all the information are available. These traces have to be analyzed in depth.

Each time a client connects to a measurement server a packet trace for this connection is started in the PCAP format [80], as shown in Figure 2.2. From these traces all the headers of each packet, incoming and outgoing, can be read. The traces are recorded on the measurement servers and transferred to a central point for processing.

Python scripts were developed to extract information from these packet traces. They run automatically for each arriving packet trace and send their output to a database. The scripts' purpose is to analyze the raw packet data to present statistics about the events happening during a TCP connection. This requires an implementation of the TCP processing of ACKs to obtain the status continuously.

The details for each event are described as follows: connection interruption in Section 3.2.2, packet reordering in Section 3.3.2, and the extraction of RTT samples in Section 4.2.

2.4 Related work

When TCP was beginning to be widely used over the new emerging wireless technologies, new challenges were arising due to the different properties compared to a wired network.

In the beginning, the prices were very high, so that the connection was switched off when not needed. This could cause problems for apps [6]. Additionally, link errors caused data loss which are unrelated to congestion. Still, TCP reduces its sending rate which caused major performance

issues [16, 81, 52, 84]. Approaches in different parts of the network were tried to compensate [15]. On the sender as a TCP congestion control algorithm [37, 83, 60], or as part of the TCP recovery [19]. It can also be handled in the network itself [7, 15], for example by modifying the TCP ACK stream [13], splitting the connection into two parts, or providing explicit loss notifications (ELN). Starting with GPRS, reliability was introduced in the wireless link itself, to locally retransmit any corrupted data. This approach was fast enough for TCP to only notice slightly higher delays instead of loss or RTO [62].

Connection interruptions can also cause problems with TCP, as these are triggering an RTO if they are too long. TCP based solutions are handling this either pro-actively by changing the flow control [39] or retroactively by reverting to the state prior to the event [59, 58]. In the network, there are approaches to handle this by splitting the connection [12].

In general, a performance enhancing proxy (PEP) in the network [23, 14] can have a wide range of functionalities to improve the TCP data transfer [46].

Problems that might arise in TCP have also been studied in simulations: rate and delay variation can cause multiple packet losses which TCP Reno cannot cope with efficiently [29], and spurious RTO due to variable rate causing delay spikes [74]. In an LTE testbed it was found [66], that an increase in RTT due to link layer retransmissions can decrease TCP performance.

In recent years the community also investigated the impact of large buffers in the network, which cause excessive latency due to the amount of buffered data. This bufferbloat issue [26, 38, 41] is quite prominent in mobile networks [48]. The issue can be tackled on the mobile device itself, by limiting the TCP RWND [49]. In the network, the queueing methods can be changed from drop-tail or drop-head to an active queue management (AQM) which is better able to control the induced latency [36, 65].

While there is a vast amount of studies on the performance an TCP and its extensions, we now focus more on 3G and 4G with TCP. Measurements have shown that from 3G HSPA onwards, mobile networks offer real broadband internet access with much improved throughput, delay, and jitter [72].

The throughput experienced by applications in mobile networks has been widely studied in literature, e.g. in 3G [44] and 4G [43], which showed that throughput is higher in 4G. These works do not list if there

might be other bottlenecks than the network capacity. While there are taxonomies for investigating the limiting factors during a connection [86], they are not widely used for mobile network.

Some papers investigate the TCP efficiency in live mobile networks based on measurements. In 3G 1xEV-DO the efficiency is between 80 % to 90 % [57] depending on the used congestion control algorithm with TCP Cubic being best. In the early years of 4G networks, the efficiency is tested by gathering data from a middlenode in the mobile operator network. They infer saturation when the throughput is above 30 Mbit/s and conclude that TCP has a utilization ratio of only 35 %. Another approach compares the TCP throughput in 3G HSPA and 4G LTE to a UDP throughput test performed right before [56]. In a stationary environment, the utilization of TCP was around 75 % for HSPA and 70 % for LTE.

For measuring and analyzing TCP, guidelines have been proposed [10]. These enable researchers to easily understand and evaluate the work of others. When testing new algorithms or a mobile network setup, it is often relied on simulations to evaluate, since it is reproducible. However, even commonly used and extensive simulators [76, 68] have their limitations [77], in that they often do not implement the required feature set for the measurement or that they cannot provide the diverse results that is potentially the case in real mobile networks. Hence, in this thesis large scale measurements in deployed mobile networks are used to investigate the performance of TCP in these networks.

2.5 Summary

This section first described mobile networks. They have grown over time to support a substantial data transfer rate with the introduction of new technologies. Nowadays, many users heavily rely on the network's availability and performance.

For measuring mobile network performance, the crowd-sourced measurement platform Netradar is introduced, and an overview of performance metrics is given. 4G networks have higher throughput and lower latency than 3G networks. But it was also visible that the performance of 4G has degraded over the last few years due to the increase in user numbers which have to share the resources.

Hence, it is of ever growing importance to make the most out of the offered resources. TCP being the dominant transport protocol in the inter-

net, it is responsible for determining the sending rate of a large fraction of data connections, and therefore plays a significant role in using these resources. The last part of the chapter described the measurement and analysis platform for TCP on the basis of Netradar.

The remainder of this work analyzes single TCP connections through mobile networks, in order to determine the current state of performance and interactions with the mobile network.

3. Delay characteristics

This chapter describes the measurement of delay characteristics for TCP connections through mobile networks. Specifically, connection interruptions and packet reordering are analyzed and found to be major sources of delay spikes. In this chapter, TCP is used as a tool to infer network characteristics.

The content is based on Publication III, Publication IV, and Publication V. The reordering detection algorithm is defined in an IETF Internet-Draft [87].

3.1 Delay during data transfer

While the previous chapter showed information on the RTT in mobile networks, this section adds to the knowledge by providing measurements of the RTT during a full bulk TCP data transfer. The performed method is not using the Netradar infrastructure, instead it is a small-scale test which aims at testing the operators in the same conditions.

3.1.1 RTT composition

In the last chapter, the RTT analyzed was based on small UDP messages. These messages traverse the network path and get delayed by several sources: propagation, transmission and processing delay.

The propagation delay is the natural limit of transmitting information, the speed of light. The further away the destination is, the more time it takes. Hence, this applies while the data is in between network nodes. The transmission delay is imposed by the capacity of the sending node. It needs time to write the data to the medium. The faster the node can send data, the less time it needs. Lastly, the processing delay is the accumulation of time the nodes on the path need to process the information inside

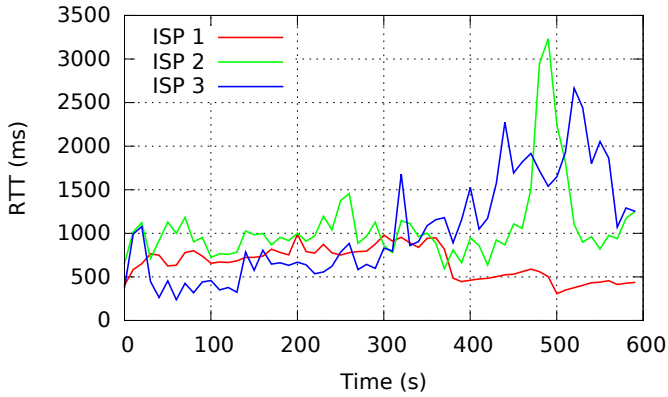


Figure 3.1. RTT progression for a measurement while stationary.

the packet. This can vary widely based on the application of the network node.

When measuring the RTT during a TCP connection, there is another source of delay which adds to the overall measured RTT and can be substantial: the queueing delay. This is shown in this section.

3.1.2 Methodology

The evaluation was performed with a server in the department and a laptop which was either stationary, or driven through the Espoo city area with a car.

Internet for the laptop was provided by three mobile phones, each with a subscriber identity module (SIM) card from a different operator. This way the networks can be tested in the same situations and at the same time. While the tests are only performed in 3G networks, the general concept and outcomes are also applicable to 4G networks.

Two bulk transfer experiments are conducted. First, the laptop remains in one location without being moved. Second, measurements are done while driving with the laptop in a car along a route of 10 km.

3.1.3 Results

The following results show the RTT behavior during a TCP connection both in stationary as well as while driving.

Stationary

The progression of the RTT during a full bulk TCP measurement in the stationary case is shown in Figure 3.1.

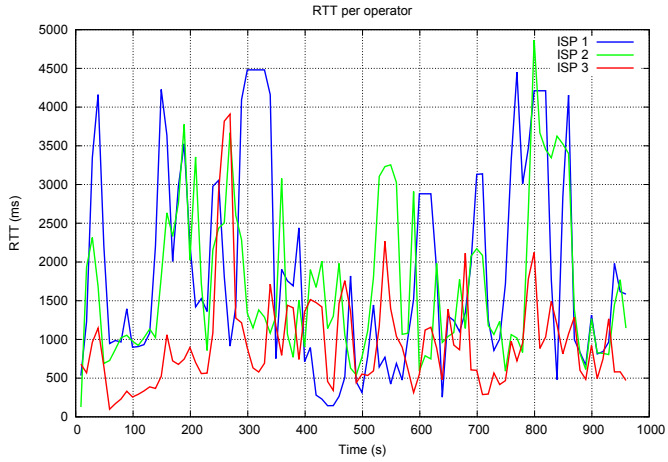


Figure 3.2. RTT progression for a measurement while driving in a car.

We see that for all three operators all values in the graph are higher than what was reported for mobile networks in the previous chapter. This shows how much different the RTT behaves when the path is under load: the data transfer itself causes the RTT to increase dramatically. Note, that the absence of the expected low values is due to the reporting interval of 1 s, after which the RTT was already increased.

Additionally, the graph shows that the maximum values are significantly higher than the minimum for each operator. While the values are fluctuating on a smaller scale, on the long run they behave more stable. For ISP 2 and 3 the values increase until around 500 s after which they drop. ISP 1 is the most stable. While the RTT also increases over time, it drops at around 400 s and 500 s. This behavior can be explained by looking at how loss based congestion control behaves: TCP increases its sending rate all the time. Only when packets are lost does it reduce its sending rate. This is what is seen as sharp drops in the graph.

In mobile networks the increase in RTT during a data transfer can be substantial. Such behavior, normally as a result from very large queues in the operator network, is called bufferbloat [38, 48, 49].

Driving

In the other test, the RTT measurement while driving in a car is performed. The results are shown in Figure 3.2. As can be seen, the results are quite different from the stationary environment. An increase over time is barely visible due to the huge variations in the RTT, which is visible for all operators.

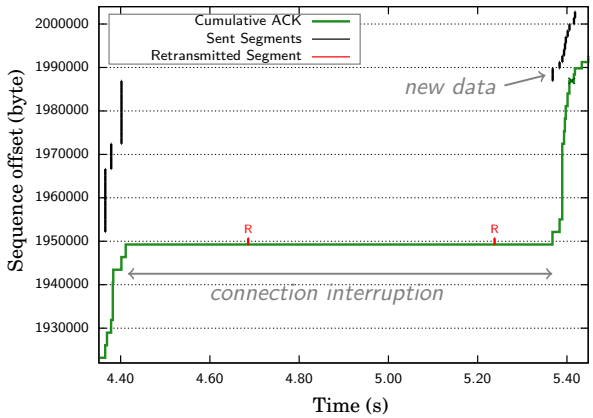


Figure 3.3. Connection interruption during a TCP connection.

The additional component that comes into play in this scenario is the high variance in throughput: while before the throughput was rather stable, here the data transmission has to cope with situations where the mobile network has to adaptively change the sending rate.

3.2 Interruptions

This section describes the measurement of end-to-end connection interruptions in connections through mobile networks and its effect on the overall performance.

3.2.1 Definition

An interruption in the end-to-end data transfer happens if no data can be transmitted between the mobile network and the mobile client, which can happen in either direction.

During a TCP connection this manifests as not receiving an ACK for a long time. This is illustrated in Figure 3.3, where the cumulative ACK remains unchanged between 4.40 s and 5.35 s, meaning that no new data gets acknowledged. As can be seen, before and after this event ACKs are arriving constantly, i.e. the cumulative ACK point increases.

3.2.2 Methodology

The difficulty in measuring interruptions lies in the way TCP sends and acknowledges data: data is sent in packets which might not correlate

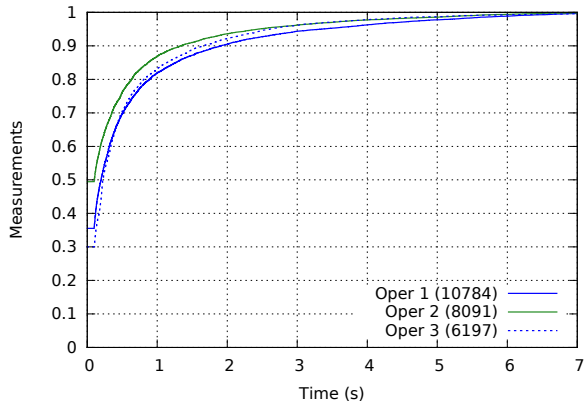


Figure 3.4. Total interruption time separated by operator (3G).

to the frames sent inside the wireless link of the mobile network, and an ACK is cumulative which might acknowledge more than one packet. These events are discrete, meaning that there is always some time passing between sending packets as well as between ACKs.

The time passing between two ACKs can vary because of sending rate and RTT variations, as a packet of data needs a certain time to be forwarded by the network. While in stable conditions it would be easy to calculate the time between ACKs from the sending rate, this is not trivial in mobile networks where the conditions may change rapidly. For example, in Figure 3.3 even the ACKs before and after the event are not evenly spaced, and we have to decide in each case if this should be handled as interruption or not.

Hence, the measurement was performed with a setup where it was more important to see how long connection interruption in mobile networks last, than to evaluate concretely where the difference between normal transmission and interruption is. A fairly high threshold of 100 ms decides if a connection interruption occurred or not.

Note, that in Figure 3.3 two RTO are triggered causing TCP to retransmit data. In this setup, the classification was made solely on the interval between two ACKs while not taking RTO retransmissions into account to have an analysis of the network, independent of the implementation details of TCP.

3.2.3 Results

To get a feeling for how severe connection interruption might be during a 10 s connection in a HSPA network, Figure 3.4 shows the behavior sepa-

Table 3.1. Impacting factors for interruptions.

	length	number	loss
Phone model	no	yes	no
Operator	no	no	yes
Time of day	no	yes	yes
User movement	no	yes	no
Signal strength	no	no	no

rated by operator as a cumulative distribution function. On the x-axis we plot the total time during connections that no data was transferred, i.e. when the connection was interrupted, and the y-axis shows the fraction of total measurements. In half of the connections interruptions are found, with almost 10 % of connections having a down time of 2 s, meaning that 20 % of the connection time no data is forwarded by the network.

The different tested factors and if they have an impact on the length of individual interruptions, number of interruptions during a connection, and the occurrence of data loss during an interruption are shown in Table 3.1. These are determined by plotting each factor to length, number, or loss, and trying to find any correlation. If none were visible the answer is no, otherwise yes. None of the factors had a significant impact on the length of individual interruptions, which suggests that this might be network congestion dependent.

The number of interruptions during a connection, i.e. the frequency of interruptions, was affected by several factors. Fast movement of the user leads to more frequent handovers which can cause a lengthy period of no data transmission. During the day there are more interruptions which is expected because more people are actively using their phones and causing network congestion. Probably the most interesting find is that some phone models cause a severe number of interruptions. One of the six most used phones had five times more interruptions than the others.

Packet losses during these events are affected by time of day and operator. The numbers for lossy interruption are 2.46 %, 0.51 %, and 0.02 %. The operator with the highest number of lossy connections had a lot of those during night time, probably due to switching off base stations. Hence, these two are related for this operator. Still, even among the other two operators we see that the difference is 25-fold.

Generally, we see that connection interruptions are not rare in 3G net-

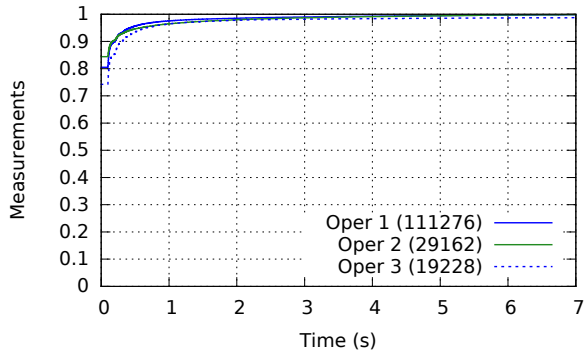


Figure 3.5. Total interruption time separated by operator in 4G networks.

works. During a significant amount of time the network is not forwarding data, which decreases the overall throughput of the connection.

In addition to the measurements in Publication III, a graph for connection interruption in 4G is presented in Figure 3.5 for the data of 2015. In comparison to the HSPA measurements in Figure 3.4, the total time of interruptions during connections has significantly decreased: only about 20 % of connections have detected interruptions, while in HSPA it was around 60 %. Reasons for this are likely a more resilient, simpler and faster core network.

3.3 Packet reordering

This section describes packet reordering in 3G and 4G mobile networks in terms of changes in delay.

3.3.1 Definition

When data is missing, the TCP receiver sends an ACK to notify the sender that data has been received, but it is not the expected piece of data. This is called out-of-order reception of data. Often, this data has been lost and must be retransmitted. However, in the case of packet reordering the same symptoms occur. Though in reality the packet is not lost but just arriving late.

Packet reordering is a network event where packets are sent out into the network in a specific order but arrive at the receiver in another. Hence, somewhere in the network the packets have been shuffled and surpassed earlier packets. An example is shown in Figure 3.6. Five packets are sent out, but packet 3 takes a different path from the others so that packet 1

and 2 arrive after packet 3.

In general, it is advised for the network to refrain from generating packet reordering. Still, it happens since the IP [69] itself is not reliable and is packet switched instead of having an assigned path. Reordering can be caused in several ways [63]:

- route changes
- load balancing on multiple paths, where the paths have different delays
- multi-core switches and routers, if a connection is handled by multiple cores
- link layer retransmissions, if the packet order is not maintained after retransmission
- prioritization queues or traffic characterization, when packets of the same connection end up in different queues

Regarding mobile networks, the otherwise rarely occurring events of route change and link layer retransmission gain importance, as packet corruption on the wireless link happens much more often than in wired networks, and users move between base stations causing route changes.

3.3.2 Methodology

Based on our general measurement setup described in Section 2.2, several TCP options are used to distinguish the cause for out-of-order arrival, which can be either packet loss or packet reordering [87]. These options are SACK and TCP Timestamps, which have been introduced in Chapter 2. Additionally, duplicate selective acknowledgments (DSACKs) [20] are used. This option extends on SACK, enabling the receiver to indicate the reception of multiple receptions of the same data.

The SACK option is only able to detect reordering if the packet has not yet been retransmitted. In such cases, the arrival of an ACK for a packet

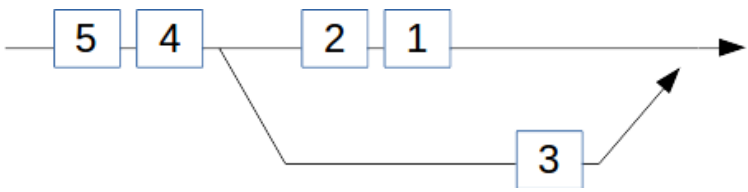


Figure 3.6. Packet reordering in the network.

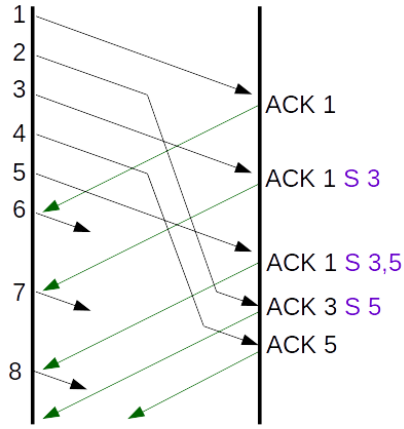


Figure 3.7. Reordering detection with SACK.

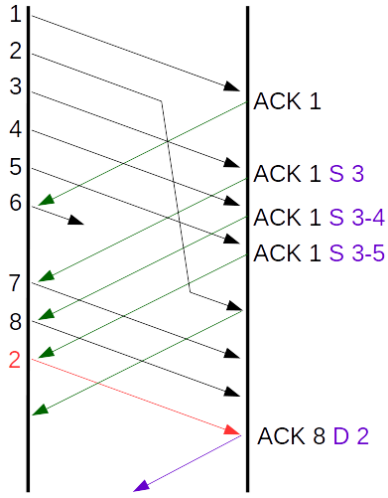


Figure 3.8. Reordering detection with DSACK.

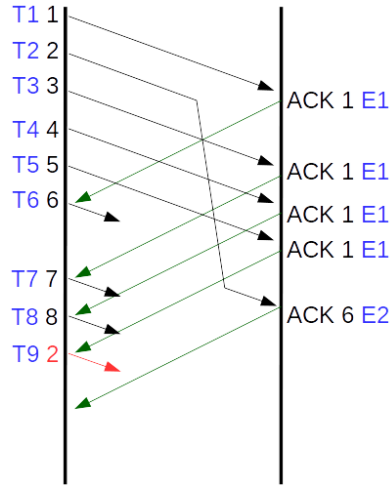


Figure 3.9. Reordering detection with timestamps.

where later sent data has been acknowledged already determines the late packet as reordered. This is shown in Figure 3.7 where packet 2 and 4 can be detecting as reordered on ACK 3 and ACK 5 respectively.

In contrast to SACK, the other two options can be used for solving the retransmission ambiguity, and therefore also work for retransmissions.

The DSACK option is sent by the receiver if a packet has been received multiple times. If the packet has been retransmitted by the sender and it is now received two times, then the retransmission was unnecessary as the original packet was still in the network. Therefore, the arrival of a DSACK at the sender marks this packet as reordered as shown in Figure 3.8. Unfortunately, the DSACK option, in contrast to Timestamps

and SACK is not negotiated. That means, that when no DSACK is ever seen on the connection it is not certain if this is because none was needed or if it is not used at all.

The TCP Timestamps option can also be used to determine the cause for out-of-order for retransmitted packets. With each packet a monotonically increasing timestamp is sent. In each ACK the newest such timestamp is echoed. Hence, if the ACK which acknowledges missing data contains the timestamp of the retransmission the data was lost. If the timestamp is lower then the original transmission was reordered. This is illustrated in Figure 3.9.

Due to these needs, we only consider connections which use both the SACK and the Timestamp option.

The reordering behavior is analyzed based on the number of reordering events during a connection, as well as the severity of each reordering. The number of reordering events is commonly given as the reordering rate, which gives the percentage of packets reordered out of the total number of packet transmitted during the connection. This is defined in RFC 4737 [63] as the Reordered Packet Ratio. For the severity there are two metrics: the reordering extent which gives the number of packets received until the reordered packet arrives, and the reordering delay which is defined as the time the packet is additionally delayed due to reordering.

One specialty of the used reordering detection methodology is, that in contrast to earlier works [18, 59, 85, 54] it is able to detect reordering even if the reordering delay is larger than the RTT. This is done by accumulating the information from timestamps and DSACK.

3.3.3 Results

The same as for connection interruptions in the previous section, for reordering there is also a visible temporal deviation as well as odd device behavior.

With one device a significant amount of additional reordering was seen, often several 100 reordering events during a 10 s connection. This might be due to faulty implementation of handling link layer retransmissions, due to which packets are handed up to higher layers in a wrong order. The reordering extent of these connections was small, only one or two packets, which would be well within the timing.

In one operator network there is also a temporal component to the amount of measured reordering. Around midnight the number of reorderings in-

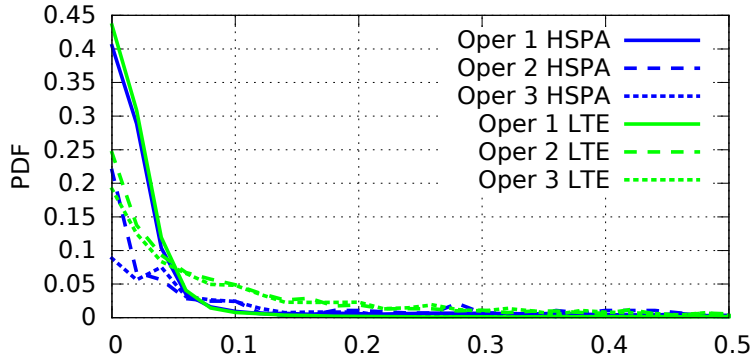


Figure 3.10. Reordering rate [%] in each tested network.

Table 3.2. Highest reordering rate [%] (95 %ile)

	Oper 1	Oper 2	Oper 3
HSPA	0.43	2.6	5.4
LTE	0.31	0.67	6.0

creases significantly and then goes back to normal two hours later. While we found a constant amount of higher reordering extents, a very high amount of small reordering extents was measured. This hints at an additional source of reordering during that time frame.

The biggest impact on reordering, though, has the operator network and even the technology. Generally, connections through 4G networks are more likely to be affected by reordering than 3G. This ranges from operator 1 with 48 % of 3G connections and 82 % of 4G connections experiencing reordering, and operator 2 between 1 % and 8 %. This might be due to the found correlation between sending rate and number of reorderings: more reorderings occur with higher sending rates.

Due to the existing correlation, Figure 3.10 shows the reordering rate for each of the tested networks. In all tested networks, the rate was generally very small for most connections, but it can reach up to 6 % in one operator network as Table 3.2 shows. The appearance of reordering throughout a connection was seen to be evenly spread out. But in the operator 1 4G network clusters were found that happened only infrequently on the order of several seconds.

The severity of the reordering shows a different picture among operators. Figure 3.11 shows the reordering extent for each reordering event. While for two of the operators most of the reordering events have a very small reordering extent, in the network of operator 2 a big part of reordered packets is bypassed by much more data. When checking the dif-

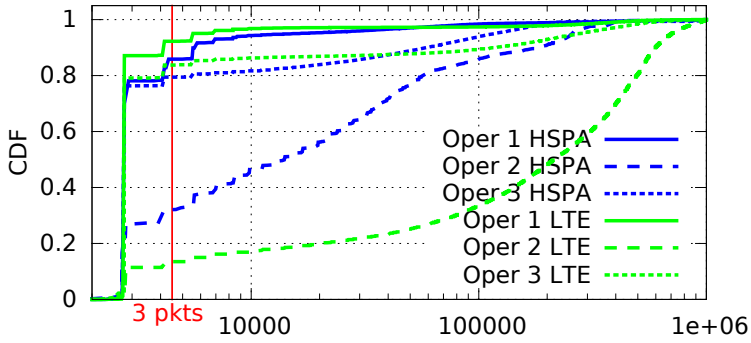


Figure 3.11. Reordering extent [B] for individual packets in each tested network.

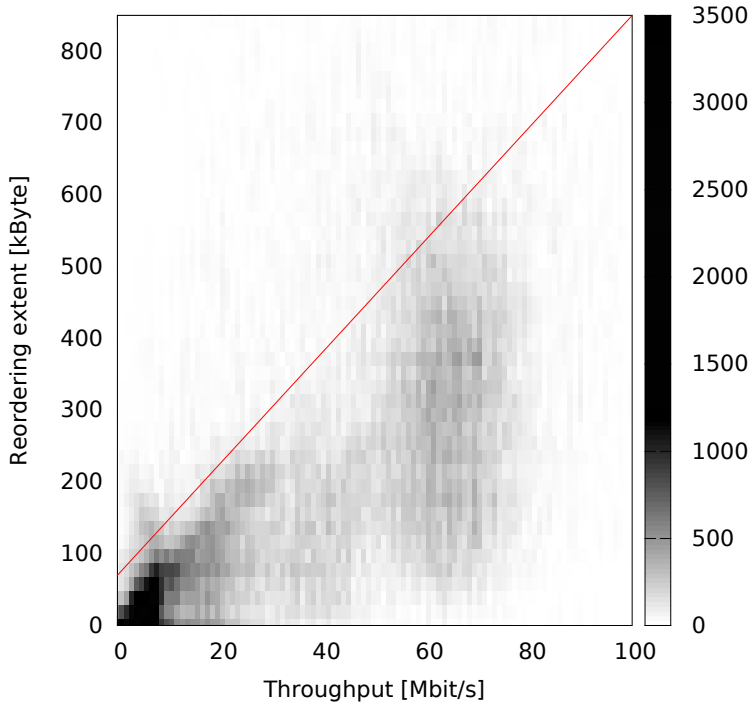


Figure 3.12. Reordering extent for individual packets over sending rate as a heat map.

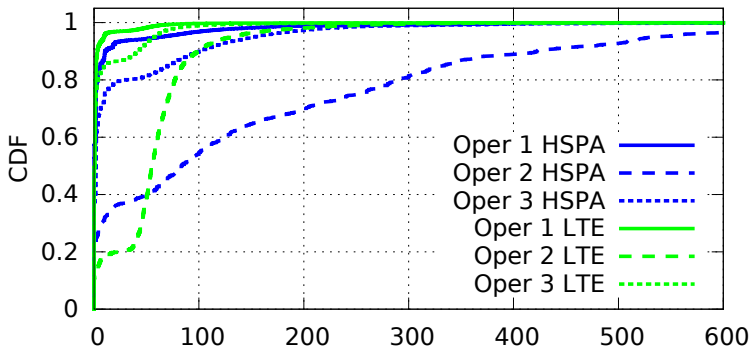


Figure 3.13. Reordering delay [ms] for individual packets in each tested network.

ference between 3G and 4G technology, it shows that in two of the networks the reordering extents are fairly similarly distributed, while in the network of operator 2 the reordering extents are bigger.

A possible explanation for bigger reordering extents in 4G networks is presented in Figure 3.12, where the reordering extent is plotted over the current sending rate. It is noticeable that in the left part of the graph very few reordering extents are high, while with higher sending rates the reordering extent is potentially higher as well. This shows that a higher sending rate leads to more packets bypassing the reordered packet.

While the reordering extent is dependent on the sending rate, the reordering delay in Figure 3.13 gives an independent overview. Again, the operator has a significant impact, with operator 1 having almost only small reordering delays and operator 2 having mostly higher reordering delays. But the distinction between 3G and 4G becomes clearer: it shows that the reordering delay for 4G networks actually decreases and levels off faster than for 3G networks. Comparing this to the behavior of the reordering extent in Figure 3.11, the assumption of sending rate dependence becomes more evident, as 4G networks have higher reordering extents while the reordering delay decreases.

3.4 Summary

This section showed what the network is providing, i.e. what the transport layer has to deal with. While previously the average delays in current networks were shown, in this section the delay during data transfer was analyzed. It was seen, that a data transfer has a substantial impact on the RTT. In mobile networks, this difference is especially visible due to big buffers that cause bufferbloat.

Additionally, reasons for delay spikes were presented and their length analyzed: connection interruptions and packet reordering.

Both characteristics suspend the transmission of data. While a connection interruption stops the transmission of any data, packet reordering delays the reception of individual pieces of data. Still, although TCP receives other data, it hands data to the application only in-order. Hence, later received data remains in the receive buffer until the missing data has arrived.

Measurements in this section showed that the two characteristics operate on different timescales. While connection interruptions appear in the

range of 100 ms to 1000 ms, packet reordering is generally a magnitude lower.

4. Protocol characteristics

This chapter describes the protocol characteristics of TCP when facing conditions present in mobile networks. While the last chapter induced network characteristics based on TCP connections, in this chapter the focus is on the protocol itself.

The first section shows how to mitigate the impact of packet reordering on the performance and reduce delay. The second section describes the current state of TCP interactions with the network.

The work is based on Publication IV and Publication VI. The reordering reaction algorithm is defined in an IETF Internet-Draft [88].

4.1 Reordering reaction

As seen in the previous chapter, packet reordering in the network causes an additional delay to some packets. In this section, an algorithm is presented which is designed to react to reordering under changing network conditions, such as present in mobile networks. Additionally, this minimizes the impact on the delay while still maintaining high throughput.

4.1.1 TCP with reordering

TCP is a reliable protocol that delivers data to the receiving application in-order. When data is missing on the receiver side, all consecutive data is buffered until the missing data is received. Only then does TCP hand all the buffered data to the application.

For each data packet that is received while data is missing, the TCP receiver sends a DUPACK, informing the sender that unexpected data has been received. The TCP sender reacts to this by retransmitting the missing data when enough DUPACKs are received and enters recovery. By default this DUPTHRESH is 3 packets [9]. In addition to retransmitting

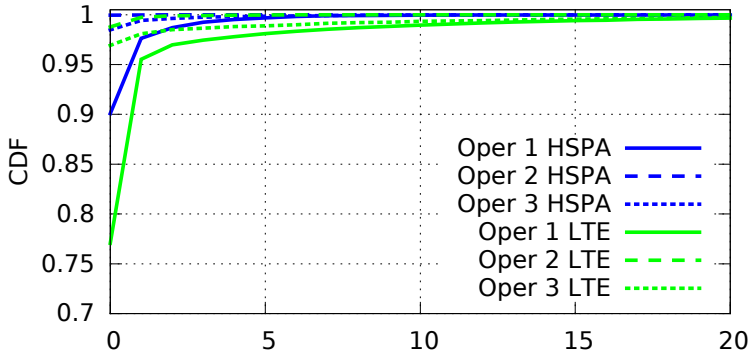


Figure 4.1. Number of spurious recoveries during a connection [#].

the data, the TCP congestion control assumes that the data is lost due to congestion in the network. Therefore, it reduces the sending rate during recovery.

When reordering is present in the network, the same symptoms are perceived: data is missing at the receiver as it is delayed by reordering and it sends DUPACKs. The sender then responds accordingly which unnecessarily retransmits data and reduces the sending rate.

Figure 3.11 in Section 3.3.3 presented the reordering extent in mobile networks. As can be seen, quite some extents are greater than the default DUPTHRESH of 3 packets, which might trigger a spurious recovery. These reordering extents might still be either clustered, i.e. causing only a single recovery event and no additional sending rate reduction, or be co-located with actual losses in which case the recovery and reduction in sending rate is needed. Hence, Figure 4.1 shows the amount of spurious recoveries caused by reordering. The graph shows that the number of connections in which the sending rate is spuriously reduced due to reordering can be quite high depending on the used network. While in the 4G network of operator 1 there are 77 % of connections without suffering from reordering, in turn it means that 23 % are.

4.1.2 Algorithm

When TCP spuriously retransmits, it performs this action since it assumes the data has been lost. Only later does it receive the information that the packet has in fact been reordered. A common approach [22, 54, 18] is therefore to delay the decision of retransmitting until all the information are available.

For cases where the packet is reordered it is optimal to just wait, as

the data is delivered to the receiver eventually. But when the data is actually lost, any waiting until the retransmission on the sender incurs an additional delay on the data delivery to the application at the receiver. In turn, this also increases the receive buffer size requirements as all intermediate data has to be stored.

The aim of a reordering reaction algorithm is therefore twofold: maintain a high throughput by ceasing to reduce the sending rate in the face of reordering, and keep the delay impact as low as possible.

The algorithm analyzed in this section, adaptive non-congestion robustness for TCP (TCP-aNCR), optimizes both aspects [88]. It uses the dependency of the reordering extent on the sending rate as shown in Section 3.3.3 to calculate the DUPTHRESH in a way that it is sending rate independent. With a changing sending rate the DUPTHRESH is then automatically adjusted, i.e. it is increased when the sending rate increases and vice versa.

In contrast to other reordering reaction algorithms like the one implemented in Linux [82] this reduces unnecessary retransmissions: when the sending rate increases also the reordering extent increases. Additionally, it reduces the delay for retransmissions when the sending rate decreases since such a high DUPTHRESH is not needed. The algorithm is based on TCP-NCR [18], which also adapts the DUPTHRESH based on the sending rate, but does not take reordering measurements into account and always waits for 1 RTT before retransmitting.

4.1.3 Methodology

TCP-aNCR is compared to the Linux reordering reaction and TCP-NCR in a controlled environment. It consists of virtual machines for sender, receiver and a middlenode. On the middlenode the network is emulated with netem [55], which can control network characteristics such as delay and reordering, and the Hierarchical Token Bucket (HTB) [34] queueing system to set the capacity of the link.

The RTT and the reordering characteristics are kept constant. The sending rate is increased and decreased throughout the lifetime of each connection to show the benefit of the TCP-aNCR algorithm.

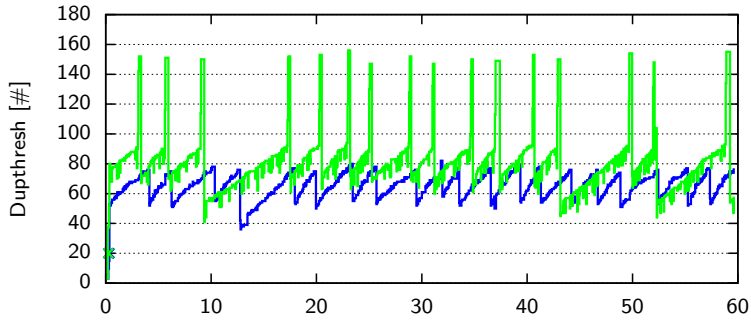


Figure 4.2. DUPTHRESH progression in stable conditions over time [s]. Green graph: TCP-NCR, blue graph: TCP-aNCR.

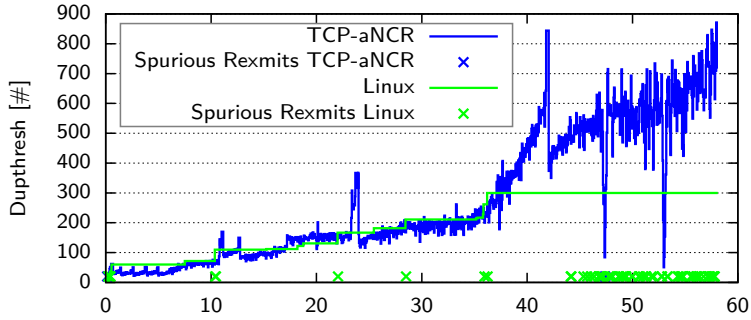


Figure 4.3. DUPTHRESH progression when capacity increases over time [s].

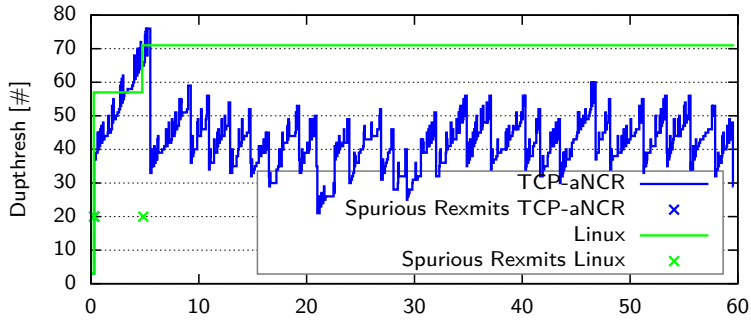


Figure 4.4. DUPTHRESH progression when capacity decreases over time [s].

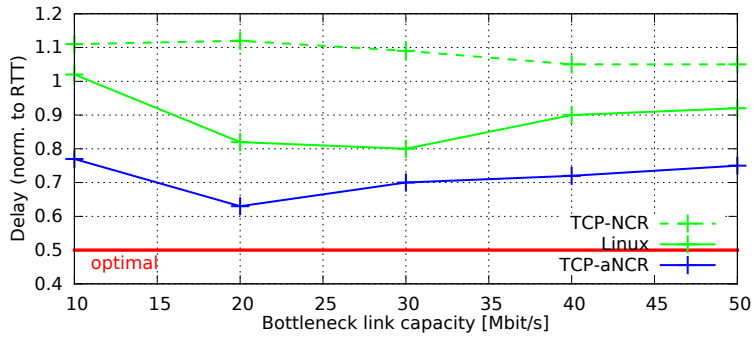


Figure 4.5. Average increase in RTT due to reordering reaction.

4.1.4 Results

The benefits of TCP-aNCR are shown in stable conditions, as well as in situations where the capacity changes.

In Figure 4.2 the approach is compared to TCP-NCR in order to show the DUPTHRESH progression over time in stable sending rate conditions. While both algorithms are robust to reordering, there is a noticeable difference: TCP-NCR shows big spikes upwards every few seconds. These are the times when packets are actually lost during congestion in the network. The high value in DUPTHRESH means that it is waited a long time before retransmitting. The proposed algorithm is as robust as TCP-NCR but by measuring the network conditions it only waits long enough to be able to decide.

The approach is also compared to Linux in Figure 4.3 for an increase in capacity and in Figure 4.4 for decreasing capacity for the flow, i.e. when another flow joins the bottleneck. In both cases Linux behaves not optimally: when the capacity increases Linux spuriously retransmits, indicated by the crosses at the bottom, since the DUPTHRESH is too low for the new higher sending rate. When the capacity decreases the DUPTHRESH stays elevated and delays retransmissions unnecessary. In both cases it is visible that the new approach combines an adaptive threshold with network measurements to provide a more favorable behavior.

Figure 4.5 shows the average increase in RTT for the reordered packet that is induced by the reordering reaction algorithm when the sending rate is halved during the connection.

The highest additional delay is seen for TCP-NCR, as it waits for 1 RTT independent of the seen reordering in the network. The Linux reordering

reaction measures reordering extents but always uses the highest measured extent as the DUPTHRESH. Therefore, it does not decrease the waiting time when the sending rate decreases and the additional delay spike stays high.

With TCP-aNCR the additional delay is the lowest throughout the tested network capacities. It reacts to the changing conditions to adapt its parameters.

4.2 TCP efficiency

This section investigates how the network, especially the queueing, interacts with the performance of TCP.

4.2.1 TCP and network queueing

When TCP sends data and the data is forwarded by the network, there are two cases than can happen: either the TCP sending rate is equal or lower than the capacity of the network, or it is higher. When the latter is the case then the data arrives faster at the bottleneck than it leaves, and the data is stored in a buffer.

This increases the RTT perceived by the sender based on the amount of buffered data. If new data arrives at the bottleneck it is stored at the end of this queue. Instead of being transmitted directly, it waits additional time until all the previous data has been sent, which is called the queueing delay. This behavior was already seen in Section 3.1.

4.2.2 Algorithm

Based on this behavior between the network and TCP, an algorithm is designed which detects the parts of a connection where it saturates the path: when the RTT is increased due to queueing of packets in the network then the path is saturated, otherwise it is not.

Algorithmically, it is checked if the RTT is higher than the lowest observed RTT, the baseRTT. Since the network, and especially the mobile network, inflicts a variation to the RTT first, the RTT is smoothed and second, a threshold is determined: if the RTT increases more than this threshold it is saturated.

RTT samples are taken for each acknowledged packet. Those samples are then smoothed with an exponentially weighted moving average

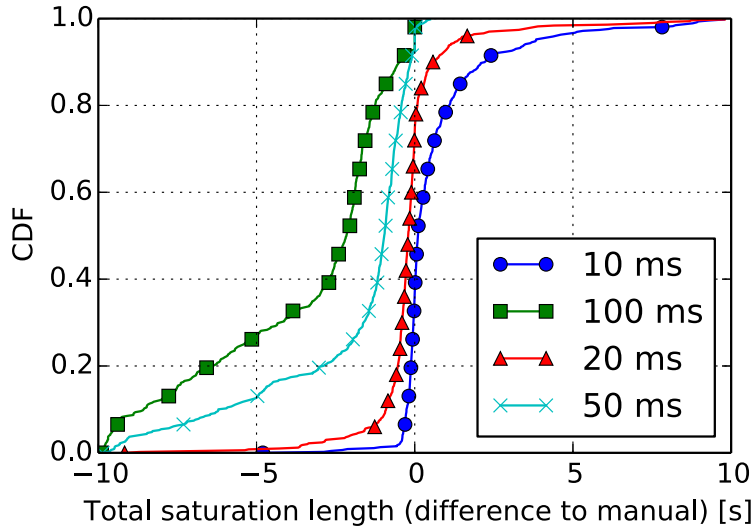


Figure 4.6. Impact of threshold choice on algorithm compared to manual tagging of 500 10 s connections.

(EWMA) with $\alpha = 1/8$ [67]. While the standard only uses one sample per RTT, this algorithm uses every sample available. This lets the EWMA adapt quicker which gives a better behavior for fast changing environments such as mobile networks.

The threshold is determined with the help of 500 Netradar connections. These are analyzed manually so that each connection has a set of time frames that are tagged with either being saturated or not. The threshold is then decided so that the algorithm provides results as close as possible to the manual tagging. This threshold is network dependent: as we have seen in the previous chapters, even among 3G and 4G networks the delay characteristics are different.

Figure 4.6 shows how the choice of threshold impacts the accuracy of the algorithm as compared to the manual tagging of 500 10 s connections. Based on this data 20 ms is taken as the threshold, as it neither overestimates nor underestimates saturation too much and provides the most accurate results. This is in line with the variation observed in 4G mobile networks [42, 53].

4.2.3 Results

This analysis of TCP behavior is based on 235 000 Netradar 10 s connections from Finnish 4G networks.

Figure 4.7 shows an overview of how much of their time the connections

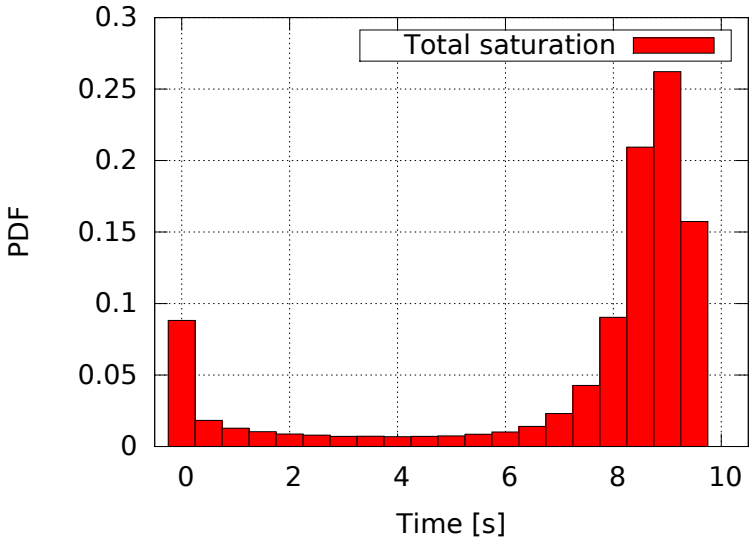


Figure 4.7. Distribution for the time spent saturated.

are saturated, i.e. use all of the offered network resources. An extensive portion resides between 8 s to 10 s, so they are saturated most of the time. The highest number of connections are saturated 9 s out of 10 s, which means they are unsaturated 10 % of the time. The graph also shows that there are about 20 % of connections which have a saturation time of less than 8 s. The following analyzes the reasons for the unsaturated parts of the connections.

There are about 8 % of connections which are never saturated. It shows that these connections experience packet loss, and hence reduce their sending rate, regularly which prevents them from increasing the speed enough. Reasons for this might be wireless link losses, although they should be rare due to wireless link retransmission schemes such as ARQ and HARQ [33], and misconfigured queueing systems which do not allow enough buffering, or resource starvation.

The most common TCP related reasons for not saturating the link is the beginning where slow start is performed. While the network probes for the network capacity the TCP sending rate is not yet high enough to fill the path. Still, in 35 % of connections there were three additional events causing unsaturated periods.

First, while no evidence was found that the RWND was limiting the throughput of any connection during congestion avoidance, there are limits on some mobile devices to reduce the amount of bufferbloat [38, 48, 49]. This can cause problems during recovery as shown in Figure 4.8. Be-

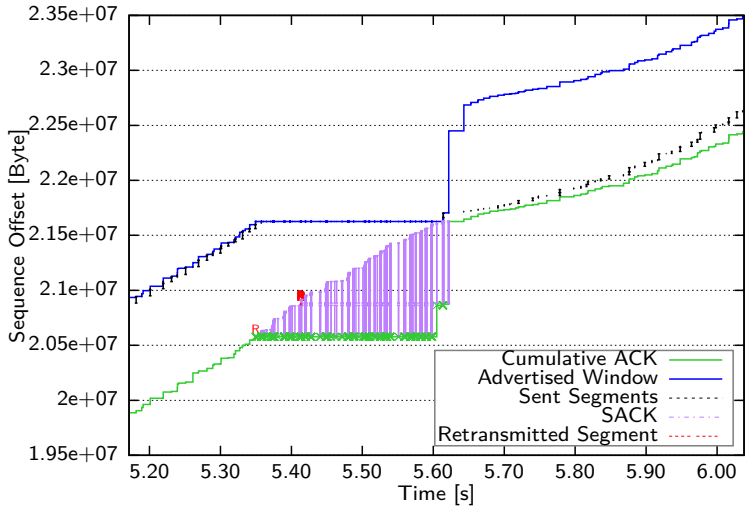


Figure 4.8. Time-sequence graph of a receiver limited TCP recovery.

fore the first DUPACK arrives at 5.35 s the sending of new data is already limited by the advertised window of the receiver. Then, during recovery, this window stays constant as the receiver has to buffer all the received data until the missing pieces have arrived. Therefore, no new data can be sent during recovery. At the end of recovery at 5.63 s almost no data is in flight anymore and the connection is unsaturated. TCP now uses slow start to achieve a high sending rate quickly.

Second, the maximum RTT allowed by the network plays an important role in the performance of TCP connections. While previously the deployed queueing mechanisms were causing significant bufferbloat, newer queueing systems such as AQM are more delay friendly and start dropping packets earlier. In general, this is a wanted behavior as this allows TCP to be more reactive. But as can be seen in Figure 4.9 it performs poorly if the maximum RTT achievable on the path is too low. It shows that TCP needs at least 200 ms in 4G networks after which the graph plateaus. The reason is that TCP congestion control algorithms such as Cubic[40] were designed for wired networks where the network capacity is stable, whereas in mobile networks it fluctuates more widely. In such cases bigger network buffers give TCP more time to increase its sending rate and catch up to the new conditions.

Hence, capacity changes are the third reason for the link not being saturated. One such case where the network capacity for the path suddenly increases is shown in Figure 4.10. The throughput is shown in green with the y-axis on the right side, the RTT is shown in red with the y-axis on

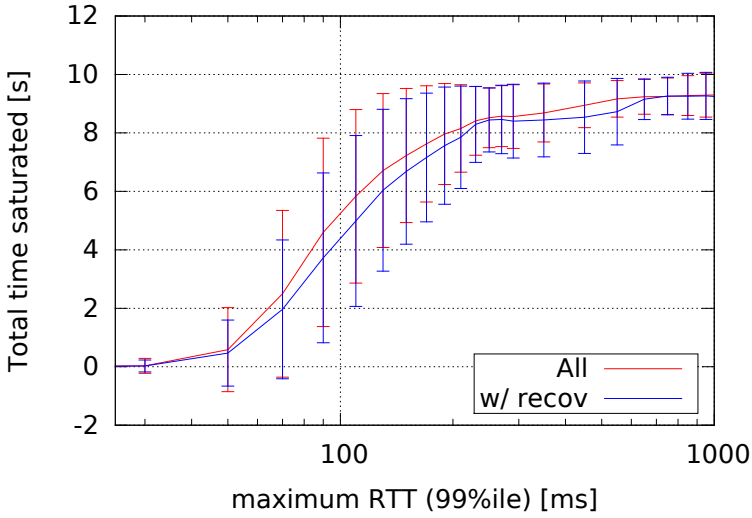


Figure 4.9. Impact of the maximum measured RTT throughout the connection.

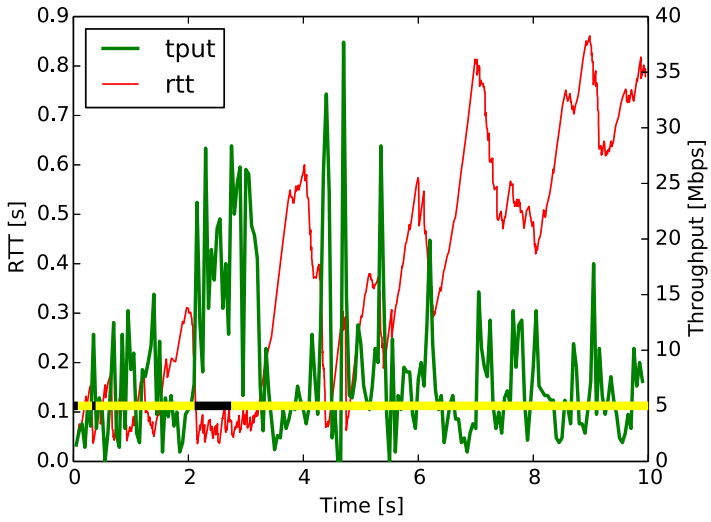


Figure 4.10. TCP is unable to react to sudden increase in network capacity. Unsaturated areas are shown in black.

the left side. The detected saturation is shown as a solid line around 5 Mbit/s with yellow indicating saturation and black unsaturated. At 2 s the capacity increases and the path is unsaturated for 500 ms. But even if the capacity suddenly decreases it can cause problems: while the network sends packets out slower, the TCP sender does not. This causes the queues in the network to fill up quickly and drop packets, and TCP reduces it sending rate. This makes the above problem more likely if the sending rate increases again.

4.3 Summary

This section analyzed the protocol characteristics of TCP with respect to the observed characteristics of mobile networks in the previous chapters.

First, a reordering reaction algorithm was presented with the goal of reacting to the scenario of mobile networks where the sending rate can vary widely. The algorithm shows to behave well without the need to acquire new samples every time the sending rate changes. It also keeps the delay for retransmissions low which reduces pressure on the receiver buffer.

Second, the interactions with the network were analyzed, showing that currently deployed mechanism on mobile phones can be counterproductive, and that low queueing delay targets are reducing throughput due to TCP's inability to adapt to changing conditions.

5. Conclusion

The previous chapters explained the motivation as well as the undertaken studies. This chapter will conclude the work. First, the answers to the research questions are detailed. Second, the results are discussed in terms of achieving good performance with TCP in modern mobile networks. Future networks are presented last.

5.1 Research questions

The research questions asked in Chapter 1 were threefold, concerning the delay characteristics in modern mobile networks, the transport layer reaction to delay spikes, and the overall performance of TCP in modern mobile networks.

These questions were studied in Finnish 3G HSPA and 4G networks, mostly on the basis of data from the Netradar measurement platform. The platform was enhanced for a deeper analysis of TCP on a packet level.

The delay characteristics were shown first: an overview was given in Chapter 2, and reasons for increase in delay were analyzed in Chapter 3. Overall, the delay is much better in 4G networks than what was seen in 3G networks by a factor of 2. The variance is generally also twice as low.

Still, connection interruption and packet reordering are seen in 3G networks as well as 4G networks, both being a major source of delay spikes in TCP connections. The amount of connection interruptions has decreased towards 4G networks, but it is still present. From a TCP perspective, it has to be handled in a way that it does not cause a spurious RTO. Appropriate handling is available and standardized: either with TCP Timestamps with Eifel, or without Timestamps according to F-RTO.

The handling of packet reordering is a different matter. As seen in Chapter 3, reordering is present in both network types. But in 4G networks we

see a higher reordering rate as well as higher reordering extents. Both of these effects are likely due to the higher sending rate. The TCP characteristics are then analyzed in Chapter 4. First, the analysis of the TCP-aNCR reordering algorithm is presented, showing it handling changes in throughput. As seen, it reacts well to changing conditions, which gives benefits in an environment like mobile networks.

Taking the delay characteristics into account, the second part of Chapter 4 then shows an algorithm to determine the efficiency of TCP in LTE networks. It was seen, that it does not operate optimally: bad interactions between optimizations and throughput changes can cause TCP to send at too low rates, wasting resources for the user and in the network.

5.2 Discussion

The question is now, what is needed for TCP to perform well in mobile networks? For 2G and early 3G networks recommendations are given in Request for Comments (RFC) 3481 [45], which are still applicable to modern networks: the window scale option [24] enables the end hosts to increase their window sizes beyond 64 kB and is necessary for high throughput environments; a high IW helps in reducing the ramp up significantly, which has lately been increased from 2 to 4 segments [9] to 10 segments [31]; path maximum transmission unit (MTU) discovery lets TCP send segments of the appropriate size for the network, as the MTU can be different in mobile networks, which reduces IP segmentation; the TCP Timestamps option [24] can be helpful for more accurate RTT measurements for retransmitted packets. Of immense importance is the TCP SACK option [61] and recovery [21] which significantly reduces the impact of a large number of lost segments. In the case of mobile networks with large queues and changing bandwidth, this happens more frequently than in wired networks.

In addition to these, we have seen in this work that connection interruption and packet reordering have to be handled carefully even in modern mobile networks. F-RTO or Eifel, and a reordering detection and reaction algorithm are therefore highly recommended when dealing with mobile networks, as the lack of either can reduce performance significantly. The reordering reaction algorithm analyzed in this work helps to deal with the frequently changing conditions in modern mobile networks when facing reordering.

What was found by studying TCP efficiency also helps to tweak performance. The slow start procedure was significantly prolonged due to delayed ACKs, as the CWND is increased only by one for each incoming ACK. This results in a ramp-up slower than the desired exponential factor of 2. The behavior can be remedied by deploying an algorithm similar to appropriate byte counting (ABC) [8] which increases the CWND based on the amount of acknowledged data.[1]

It was also seen, that there are receiver side limitations to the RWND. Those are in place to reduce the impact of bufferbloat on the RTT and increase responsiveness of TCP. However, it was visible that these optimizations can cause problems during recovery: when the sending of new data is limited by the RWND on recovery entry, no new data can be sent. This leaves TCP starved for data at the end of recovery and slow start is invoked again. There are several possible solutions for this problem: (a) on the receiver side: optimize the algorithm to increase the window when data is missing (b) on the sender side: introduce delay awareness in the congestion control which makes receiver side algorithms obsolete, and (c) in the network: reduce the amount of queueing. While a receiver side solution seems to be the easiest, it splits control of the sending rate to two parties. As we have seen, this caused problems already and might therefore not be the best solution. A sender side algorithm is very complex to achieve, as it would require a replacement of the congestion control scheme.

The network solution would tackle the problem where it arises and would be the clean solution. Many operators are aware of the bufferbloat issue, and more networks are updating the queueing. Most commonly, the replacement is an AQM. There are several benefits for operators as well as users [11], for example, the number of dropped packets is reduced which makes TCP recovery more efficient, and a decrease in delay which is the concern when tackling bufferbloat. For operators, it is also important to reduce the amount of queued data in the network, not only because it overuses the systems resources, but also to reduce the amount of stale data when the transmission is interrupted. The down side of this improved scheme was seen when studying the TCP efficiency, which showed a clear trend for lower efficiency when the maximum delay was lower. With longer queues TCP has more time to catch up, while improved

[1] In the Linux kernel this algorithm was superseded by a new one, but the principle remains the same.

queueing provides less time.

5.3 Future networks and algorithms

When using the TCP enhancements presented above, the main problem that remains is TCP's inability to handle the variable throughput when using loss based congestion control such as Reno [9] or Cubic [40].

When serving only customers from mobile networks, a delay based approach might be suitable as a sender only solution, as the queueing at the network bottleneck is mostly per customer. But in general, servers of content providers do not have the knowledge from which network the customer requests data. In such cases, a purely delay based approach has the disadvantage of achieving a much lower throughput when competing with loss based approaches.

As such, help from the network or the receiver can be advantageous. A very common network based solution to congestion is explicit congestion notification (ECN) [73], which flags packets when congestion is imminent without dropping them. This information is then carried back to the sender to inform him about congestion in the network. For example, Datacenter TCP [17] makes use of this information to achieve high throughput and low latency in a datacenter environment. Still, it only handles the reaction to congestion, and does not increase more aggressively when resources are free. Additionally, in contrast to a datacenter, it cannot be assumed that all routers on the path implement ECN. Therefore, a lost packet has to be seen as congestion and appropriate measures must be taken. Another proposed network based solution is Mobile Throughput Guidance [47], where the network appends a TCP option to packets to indicate the network capacity to the sender. However, discussions show that, despite their usefulness in the mobile network, its implications on the internet are unclear. For one, such a solution assumes that the bottleneck is in the mobile network, which can be devastating for the rest of the path.

A recent solution for media traffic might be considered useful, where the receiver sends reports to the sender. These can include several metrics, including amount of received bytes, losses, and delay [78]. This information can be used to improve current congestion control, while maintaining full control on the sender side.

In the future, 5th generation (5G) networks will supersede the exist-

ing 4G technology. For the end user, it promises higher throughput and lower delays. Lower delays are to be achieved by sub millisecond link layer latency, lower network processing delays, and potentially AQM. In order to achieve even higher throughput on the available spectrum, the usage of massive multiple-input multiple-output (MIMO) and beamforming is promising. Especially the higher throughput for customers might increase the issues seen above, as they did from 3G to 4G networks in this study. A higher maximum throughput might lead to even bigger fluctuations of the throughput inside a connection. This makes it even more important to achieve a high interoperability of the mobile network and the end nodes.

References

[1] 3rd Generation Partnership Project (3GPP). *GPRS & EDGE*. URL: http://www.3gpp.org/gprs-edge (visited on 02/08/2017).

[2] 3rd Generation Partnership Project (3GPP). *HSPA*. URL: http://www.3gpp.org/hspa (visited on 02/08/2017).

[3] 3rd Generation Partnership Project (3GPP). *LTE*. URL: http://www.3gpp.org/lte (visited on 02/08/2017).

[4] 3rd Generation Partnership Project (3GPP). *LTE-Advanced*. URL: http://www.3gpp.org/lte-advanced (visited on 02/08/2017).

[5] 3rd Generation Partnership Project (3GPP). *UMTS*. URL: http://www.3gpp.org/umts (visited on 02/08/2017).

[6] T. Alanko, M. Kojo, H. Laamanen, M. Liljeberg, M. Moilanen, and K. Raatikainen. "Measured Performance of Data Transmission over Cellular Telephone Networks". In: *SIGCOMM Comput. Commun. Rev.* 24.5 (1994), pp. 24–44. DOI: 10.1145/205511.205513.

[7] T. Alanko, M. Kojo, H. Laamanen, K. Raatikainen, and T. M. "Mobile computing based on GSM: The Mowgli approach". In: *IFIP'96 World Conference – Mobile Communications*. Springer, 1996.

[8] M. Allman. *TCP Congestion Control with Appropriate Byte Counting (ABC)*. RFC 3465 (Experimental). Internet Engineering Task Force, Feb. 2003. URL: http://www.ietf.org/rfc/rfc3465.txt.

[9] M. Allman, V. Paxson, and E. Blanton. *TCP Congestion Control*. RFC 5681 (Draft Standard). Internet Engineering Task Force, Sept. 2009. URL: http://www.ietf.org/rfc/rfc5681.txt.

[10] L. Andrew, C. Marcondes, S. Floyd, L. Dunn, R. Guillier, W. Gang, L. Eggert, S. Ha, and I. Rhee. "Towards a common TCP evaluation suite". In: *Proceedings of the International Workshop on Protocols for Fast Long-Distance Networks (PFLDnet)*. 2008.

[11] F. Baker and G. Fairhurst. *IETF Recommendations Regarding Active Queue Management*. RFC 7567 (Best Current Practice). Internet Engineering Task Force, July 2015. URL: http://www.ietf.org/rfc/rfc7567.txt.

[12] A. Bakre and B. R. Badrinath. “I-TCP: indirect TCP for mobile hosts”. In: *Proceedings of 15th International Conference on Distributed Computing Systems*. 1995, pp. 136–143. DOI: 10.1109/ICDCS.1995.500012.

[13] H. Balakrishnan, S. Seshan, E. Amir, and R. H. Katz. “Improving TCP/IP Performance over Wireless Networks”. In: *Proceedings of the 1st Annual International Conference on Mobile Computing and Networking*. ACM, 1995, pp. 2–11. DOI: 10.1145/215530.215544.

[14] H. Balakrishnan, V. Padmanabhan, G. Fairhurst, and M. Sooriyabandara. *TCP Performance Implications of Network Path Asymmetry*. RFC 3449 (Best Current Practice). Internet Engineering Task Force, Dec. 2002. URL: http://www.ietf.org/rfc/rfc3449.txt.

[15] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz. “A comparison of mechanisms for improving TCP performance over wireless links”. In: *IEEE/ACM Transactions on Networking* 5.6 (1997), pp. 756–769. DOI: 10.1109/90.650137.

[16] C. Barakat, E. Altman, and W. Dabbous. “On TCP performance in a heterogeneous network: a survey”. In: *IEEE Communications Magazine* 38.1 (2000), pp. 40–46. DOI: 10.1109/35.815451.

[17] S. Bensley, L. Eggert, D. Thaler, P. Balasubramanian, G. Judd, and M. Stanley. *Datacenter TCP (DCTCP): TCP Congestion Control for Datacenters*. Internet-Draft. Internet Engineering Task Force, Feb. 2017. URL: http://tools.ietf.org/html/draft-ietf-tcpm-dctcp.

[18] S. Bhandarkar, A. L. N. Reddy, M. Allman, and E. Blanton. *Improving the Robustness of TCP to Non-Congestion Events*. RFC 4653 (Experimental). Internet Engineering Task Force, Aug. 2006. URL: http://www.ietf.org/rfc/rfc4653.txt.

[19] S. Bhandarkar, N. E. Sadry, A. L. N. Reddy, and N. H. Vaidya. “TCP-DCR: a novel protocol for tolerating wireless channel errors”. In: *IEEE Transactions on Mobile Computing* 4.5 (2005), pp. 517–529. DOI: 10.1109/TMC.2005.72.

[20] E. Blanton and M. Allman. *Using TCP Duplicate Selective Acknowledgement (DSACKs) and Stream Control Transmission Protocol (SCTP) Duplicate Transmission Sequence Numbers (TSNs) to Detect Spurious Retransmissions*. RFC 3708 (Experimental). Internet Engineering Task Force, Feb. 2004. URL: http://www.ietf.org/rfc/rfc3708.txt.

[21] E. Blanton, M. Allman, L. Wang, I. Jarvinen, M. Kojo, and Y. Nishida. *A Conservative Loss Recovery Algorithm Based on Selective Acknowledgment (SACK) for TCP*. RFC 6675 (Proposed Standard). Internet Engineering Task Force, Aug. 2012. URL: http://www.ietf.org/rfc/rfc6675.txt.

[22] E. Blanton and M. Allman. “On Making TCP More Robust to Packet Reordering”. In: *ACM SIGCOMM Comput. Commun. Rev.* 32.1 (2002), pp. 20–30. DOI: 10.1145/510726.510728.

[23] J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby. *Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations*. RFC 3135 (Informational). Internet Engineering Task Force, June 2001. URL: http://www.ietf.org/rfc/rfc3135.txt.

[24] D. Borman, B. Braden, V. Jacobson, and R. Scheffenegger. *TCP Extensions for High Performance*. RFC 7323 (Proposed Standard). Internet Engineering Task Force, Sept. 2014. URL: http://www.ietf.org/rfc/rfc7323.txt.

[25] R. Braden. *Requirements for Internet Hosts - Communication Layers*. RFC 1122 (INTERNET STANDARD). Updated by RFCs 1349, 4379, 5884, 6093, 6298, 6633, 6864. Internet Engineering Task Force, Oct. 1989. URL: http://www.ietf.org/rfc/rfc1122.txt.

[26] *Bufferbloat.net*. 2017. URL: https://www.bufferbloat.net/ (visited on 09/10/2017).

[27] Canonical Ltd. *Ubuntu Linux operating system*. 2016. URL: http://www.ubuntu.com/ (visited on 11/28/2016).

[28] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson. “BBR: Congestion-Based Congestion Control”. In: *ACM Queue* 14.5 (2016), 50:20–50:53. DOI: 10.1145/3012426.3022184.

[29] M. C. Chan and R. Ramjee. “TCP/IP Performance over 3G Wireless Links with Rate and Delay Variation”. In: *Proceedings of the 8th Annual International Conference on Mobile Computing and Networking*. ACM, 2002, pp. 71–82. DOI: 10.1145/570645.570655.

[30] Y. Cheng and N. Cardwell. *RACK: a time-based fast loss detection algorithm for TCP*. Internet-Draft. Internet Engineering Task Force, July 2016. URL: http://tools.ietf.org/html/draft-cheng-tcpm-rack.

[31] J. Chu, N. Dukkipati, Y. Cheng, and M. Mathis. *Increasing TCP’s Initial Window*. RFC 6928 (Experimental). Internet Engineering Task Force, Apr. 2013. URL: http://www.ietf.org/rfc/rfc6928.txt.

[32] Cisco Systems. *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2015–2020 White Paper*. 2016. URL: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html (visited on 12/20/2016).

[33] E. Dahlman, S. Parkvall, J. Skold, and P. Beming. *3G Evolution; HSPA and LTE for Mobile Broadband*. Academic Press, 2007. ISBN: 9780123725332.

[34] M. Devera. *Hierarchy Token Bucket*. 2016. URL: http://lartc.org/manpages/tc-htb.html (visited on 11/09/2016).

[35] M. Duke, R. Braden, W. Eddy, E. Blanton, and A. Zimmermann. *A Roadmap for Transmission Control Protocol (TCP) Specification Documents*. RFC 7414 (Informational). Updated by RFC 7805. Internet Engineering Task Force, Feb. 2015. URL: http://www.ietf.org/rfc/rfc7414.txt.

[36] S. Floyd and V. Jacobson. "Random early detection gateways for congestion avoidance". In: *IEEE/ACM Transactions on Networking* 1.4 (1993), pp. 397–413. DOI: 10.1109/90.251892.

[37] C. P. Fu and S. C. Liew. "TCP Veno: TCP enhancement for transmission over wireless access networks". In: *IEEE Journal on Selected Areas in Communications* 21.2 (2003), pp. 216–228. DOI: 10.1109/JSAC.2002.807336.

[38] J. Gettys and K. Nichols. "Bufferbloat: Dark Buffers in the Internet". In: *ACM Queue* 9.11 (2011), 40:40–40:54. DOI: 10.1145/2063166.2071893.

[39] T. Goff, J. Moronski, D. S. Phatak, and V. Gupta. "Freeze-TCP: a true end-to-end TCP enhancement mechanism for mobile environments". In: *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies*. Vol. 3. 2000, pp. 1537–1545. DOI: 10.1109/INFCOM.2000.832552.

[40] S. Ha, I. Rhee, and L. Xu. "CUBIC: A New TCP-friendly High-speed TCP Variant". In: *ACM SIGOPS Oper. Syst. Rev.* 42.5 (2008), pp. 64–74. DOI: 10.1145/1400097.1400105.

[41] T. Høiland-Jørgensen, B. Ahlgren, P. Hurtig, and A. Brunstrom. "Measuring Latency Variation in the Internet". In: *Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies*. CoNEXT '16. ACM, 2016, pp. 473–480. DOI: 10.1145/2999572.2999603.

[42] H. Holma and A. Toskala. *LTE for UMTS: Evolution to LTE-advanced*. John Wiley & Sons, 2011. ISBN: 0470660007, 9780470660003.

[43] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck. "A Close Examination of Performance and Power Characteristics of 4G LTE Networks". In: *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*. ACM, 2012, pp. 225–238. DOI: 10.1145/2307636.2307658.

[44] J. Huang, Q. Xu, B. Tiwana, Z. M. Mao, M. Zhang, and P. Bahl. "Anatomizing Application Performance Differences on Smartphones". In: *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*. ACM, 2010, pp. 165–178. DOI: 10.1145/1814433.1814452.

[45] H. Inamura, G. Montenegro, R. Ludwig, A. Gurtov, and F. Khafizov. *TCP over Second (2.5G) and Third (3G) Generation Wireless Networks*. RFC 3481 (Best Current Practice). Internet Engineering Task Force, Feb. 2003. URL: http://www.ietf.org/rfc/rfc3481.txt.

[46] M. Ivanovich, P. W. Bickerdike, and J. C. Li. "On TCP performance enhancing proxies in a wireless environment". In: *IEEE Communications Magazine* 46.9 (2008), pp. 76–83. DOI: 10.1109/MCOM.2008.4623710.

[47] A. Jain, A. Terzis, H. Flinck, N. Sprecher, S. Arunachalam, and K. Smith. *Mobile Throughput Guidance Inband Signaling Protocol*. Internet-Draft. Internet Engineering Task Force, Sept. 2015. URL: http://tools.ietf.org/html/draft-flinck-mobile-throughput-guidance.

[48] H. Jiang, Z. Liu, Y. Wang, K. Lee, and I. Rhee. "Understanding Bufferbloat in Cellular Networks". In: *Proceedings of the 2012 ACM SIGCOMM Workshop on Cellular Networks: Operations, Challenges, and Future Design*. ACM, 2012, pp. 1–6. DOI: 10.1145/2342468.2342470.

[49] H. Jiang, Y. Wang, K. Lee, and I. Rhee. "Tackling Bufferbloat in 3G/4G Networks". In: *Proceedings of the 2012 ACM Conference on Internet Measurement Conference*. ACM, 2012, pp. 329–342. DOI: 10.1145/2398776.2398810.

[50] P. Karn and C. Partridge. "Improving Round-trip Time Estimates in Reliable Transport Protocols". In: *ACM SIGCOMM Comput. Commun. Rev.* 17.5 (1987), pp. 2–7. DOI: 10.1145/55483.55484.

[51] S. S. Krishnan and R. K. Sitaraman. "Video Stream Quality Impacts Viewer Behavior: Inferring Causality Using Quasi-experimental Designs". In: *Proceedings of the 2012 Internet Measurement Conference*. ACM, 2012, pp. 211–224. DOI: 10.1145/2398776.2398799.

[52] A. Kumar. "Comparative performance analysis of versions of TCP in a local network with a lossy link". In: *IEEE/ACM Transactions on Networking* 6.4 (1998), pp. 485–498. DOI: 10.1109/90.720921.

[53] M. Laner, P. Svoboda, P. Romirer-Maierhofer, N. Nikaein, F. Ricciato, and M. Rupp. "A comparison between one-way delays in operating HSPA and LTE networks". In: *10th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)*. IEEE, 2012, pp. 286–292.

[54] K. C. Leung and C. Ma. "Enhancing TCP performance to persistent packet reordering". In: *IEEE KICS Journal of Communications and Networks* 7.3 (2005), pp. 385–393. DOI: 10.1109/JCN.2005.6389822.

[55] Linux foundation. *netem*. 2016. URL: http://www.linuxfoundation.org/collaborate/workgroups/networking/netem (visited on 11/09/2016).

[56] K. Liu and J. Y. Lee. “On Improving TCP Performance over Mobile Data Networks”. In: *IEEE Transactions on Mobile Computing* 15.10 (2016), pp. 2522–2536. DOI: 10.1109/TMC.2015.2500227.

[57] X. Liu, A. Sridharan, S. Machiraju, M. Seshadri, and H. Zang. “Experiences in a 3G Network: Interplay Between the Wireless Channel and Applications”. In: *Proceedings of the 14th ACM International Conference on Mobile Computing and Networking*. ACM, 2008, pp. 211–222. DOI: 10.1145/1409944.1409969.

[58] R. Ludwig and A. Gurtov. *The Eifel Response Algorithm for TCP*. RFC 4015 (Proposed Standard). Internet Engineering Task Force, Feb. 2005. URL: http://www.ietf.org/rfc/rfc4015.txt.

[59] R. Ludwig and M. Meyer. *The Eifel Detection Algorithm for TCP*. RFC 3522 (Experimental). Internet Engineering Task Force, Apr. 2003. URL: http://www.ietf.org/rfc/rfc3522.txt.

[60] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Wang. “TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links”. In: *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*. ACM, 2001, pp. 287–297. DOI: 10.1145/381677.381704.

[61] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. *TCP Selective Acknowledgment Options*. RFC 2018 (Proposed Standard). Internet Engineering Task Force, Oct. 1996. URL: http://www.ietf.org/rfc/rfc2018.txt.

[62] M. Meyer. “TCP performance over GPRS”. In: *IEEE Wireless Communications and Networking Conference*. IEEE, 1999, pp. 1248–1252. DOI: 10.1109/WCNC.1999.796937.

[63] A. Morton, L. Ciavattone, G. Ramachandran, S. Shalunov, and J. Perser. *Packet Reordering Metrics*. RFC 4737 (Proposed Standard). Updated by RFC 6248. Internet Engineering Task Force, Nov. 2006. URL: http://www.ietf.org/rfc/rfc4737.txt.

[64] *Netradar*. 2017. URL: http://www.netradar.org (visited on 02/08/2017).

[65] K. Nichols and V. Jacobson. “Controlling Queue Delay”. In: *Commun. ACM* 55.7 (2012), pp. 42–50. DOI: 10.1145/2209249.2209264.

[66] H.-S. Park, J.-Y. Lee, and B.-C. Kim. “TCP performance issues in LTE networks”. In: *International Conference on ICT Convergence (ICTC)*. IEEE, 2011, pp. 493–496. DOI: 10.1109/ICTC.2011.6082645.

[67] V. Paxson, M. Allman, J. Chu, and M. Sargent. *Computing TCP’s Retransmission Timer*. RFC 6298 (Proposed Standard). Internet Engineering Task Force, June 2011. URL: http://www.ietf.org/rfc/rfc6298.txt.

[68] G. Piro, N. Baldo, and M. Miozzo. “An LTE Module for the Ns-3 Network Simulator”. In: *Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques (SIMUTools ’11)*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2011, pp. 415–422.

[69] J. Postel. *Internet Protocol*. RFC 791 (INTERNET STANDARD). Updated by RFCs 1349, 2474, 6864. Internet Engineering Task Force, Sept. 1981. URL: http://www.ietf.org/rfc/rfc791.txt.

[70] J. Postel. *Transmission Control Protocol*. RFC 793 (INTERNET STANDARD). Updated by RFCs 1122, 3168, 6093, 6528. Internet Engineering Task Force, Sept. 1981. URL: http://www.ietf.org/rfc/rfc793.txt.

[71] J. Postel. *User Datagram Protocol*. RFC 768 (INTERNET STANDARD). Internet Engineering Task Force, Aug. 1980. URL: http://www.ietf.org/rfc/rfc768.txt.

[72] J. Prokkola, P. H. J. Perala, M. Hanski, and E. Piri. “3G/HSPA Performance in Live Networks from the End User Perspective”. In: *2009 IEEE International Conference on Communications*. 2009, pp. 1–6. DOI: 10.1109/ICC.2009.5198575.

[73] K. Ramakrishnan, S. Floyd, and D. Black. *The Addition of Explicit Congestion Notification (ECN) to IP*. RFC 3168 (Proposed Standard). Updated by RFCs 4301, 6040. Internet Engineering Task Force, Sept. 2001. URL: http://www.ietf.org/rfc/rfc3168.txt.

[74] F. Ren and C. Lin. “Modeling and Improving TCP Performance over Cellular Link with Variable Bandwidth”. In: *IEEE Transactions on Mobile Computing* 10.8 (2011), pp. 1057–1070. DOI: 10.1109/TMC.2010.234.

[75] I. Rhee, L. Xu, S. Ha, A. Zimmermann, L. Eggert, and R. Scheffenegger. *CUBIC for Fast Long-Distance Networks*. Internet-Draft. Internet Engineering Task Force, Feb. 2017. URL: http://tools.ietf.org/html/draft-ietf-tcpm-cubic.

[76] G. F. Riley and T. R. Henderson. “The ns-3 Network Simulator”. In: *Modeling and Tools for Network Simulation*. Ed. by K. Wehrle, M. Güneş, and J. Gross. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 15–34. DOI: 10.1007/978-3-642-12331-3_2.

[77] M. H. Rosbach. *Verification of Network Simulators : The good, the bad and the ugly*. Master thesis. Nov. 2012. URL: http://urn.nb.no/URN:NBN:no-33640.

[78] Z. Sarker, C. Perkins, V. Singh, and M. Ramalho. *RTP Control Protocol (RTCP) Feedback for Congestion Control*. Internet-Draft. Internet Engineering Task Force, Oct. 2016. URL: http://tools.ietf.org/html/draft-dt-rmcat-feedback-message.

[79] P. Sarolahti, M. Kojo, K. Yamamoto, and M. Hata. *Forward RTO-Recovery (F-RTO): An Algorithm for Detecting Spurious Retransmission Timeouts with TCP*. RFC 5682 (Proposed Standard). Internet Engineering Task Force, Sept. 2009. URL: http://www.ietf.org/rfc/rfc5682.txt.

[80] Tcpdump Developers. *tcpdump/libpcap*. URL: http://www.tcpdump.org (visited on 11/28/2016).

[81] Y. Tian, K. Xu, and N. Ansari. "TCP in wireless environments: problems and solutions". In: *IEEE Communications Magazine* 43.3 (2005), S27–S32. DOI: 10.1109/MCOM.2005.1404595.

[82] L. Torvalds. *The Linux Kernel*. 2016. URL: http://www.kernel.org/ (visited on 11/09/2016).

[83] K. Xu, Y. Tian, and N. Ansari. "TCP-Jersey for wireless IP communications". In: *IEEE Journal on Selected Areas in Communications* 22.4 (2004), pp. 747–756. DOI: 10.1109/JSAC.2004.825989.

[84] G. Xylomenos, G. C. Polyzos, P. Mahonen, and M. Saaranen. "TCP performance issues over wireless links". In: *IEEE Communications Magazine* 39.4 (2001), pp. 52–58. DOI: 10.1109/35.917504.

[85] M. Zhang, B. Karp, S. Floyd, and L. Peterson. "RR-TCP: A Reordering-Robust TCP with DSACK". In: *Proceedings of the 11th IEEE International Conference on Network Protocols (ICNP '03)*. IEEE Computer Society, 2003, pp. 95–.

[86] Y. Zhang, L. Breslau, V. Paxson, and S. Shenker. "On the Characteristics and Origins of Internet Flow Rates". In: *ACM SIGCOMM Comput. Commun. Rev.* 32.4 (2002), pp. 309–322. DOI: 10.1145/964725.633055.

[87] A. Zimmermann, L. Schulte, C. Wolff, and A. Hannemann. *Detection and Quantification of Packet Reordering with TCP*. Internet-Draft. Internet Engineering Task Force, Nov. 2014. URL: http://tools.ietf.org/html/draft-zimmermann-tcpm-reordering-detection.

[88] A. Zimmermann, L. Schulte, C. Wolff, and A. Hannemann. *Making TCP Adaptively Robust to Non-Congestion Events*. Internet-Draft. Internet Engineering Task Force, Nov. 2014. URL: http://tools.ietf.org/html/draft-zimmermann-tcpm-reordering-reaction.

Aalto-DD 244/2017

ISBN 978-952-60-7761-1 (printed)
ISBN 978-952-60-7762-8 (pdf)
ISSN-L 1799-4934
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)

Aalto University
School of Electrical Engineering
Department of Communications and Networking
www.aalto.fi

BUSINESS +
ECONOMY

ART +
DESIGN +
ARCHITECTURE

SCIENCE +
TECHNOLOGY

CROSSOVER

**DOCTORAL
DISSERTATIONS**