Quality of Service and Dependability of Cellular Vehicular Communication Networks

Von der Fakultät für Elektrotechnik und Informatik
der Gottfried Wilhelm Leibniz Universität Hannover
zur Erlangung des akademischen Grades

Doktor-Ingenieur

genehmigte

Dissertation

von

M.Sc. Mark Akselrod
geboren am 18. Oktober 1989 in St. Petersburg

2023

Referent	:	Prof. Dr.-Ing. Markus Fidler
Korreferent	:	Prof. Dr.-Ing. Amr Rizk
Tag der Promotion	:	12. April 2023

ABSTRACT

Improving the dependability of mobile network applications is a complicated task for many reasons: Especially in Germany, the development of cellular infrastructure has not always been fast enough to keep up with the growing demand, resulting in many blind spots that cause communication outages. However, even when the infrastructure is available, the mobility of the users still poses a major challenge when it comes to the dependability of applications: As the user moves, the capacity of the channel can experience major changes. This can mean that applications like adjustable bitrate video streaming cannot infer future performance by analyzing past download rates, as it will only have old information about the data rate at a different location.

In this work, we explore the use of 4G LTE for dependable communication in mobile vehicular scenarios. For this, we first look at the performance of LTE, especially in mobile environments, and how it has developed over time. We compare measurements performed several years apart and look at performance differences in urban and rural areas. We find that even though the continued development of the 4G standard has enabled better performance in theory, this has not always been reflected in real-life performance due to the slow development of infrastructure, especially along highways.

We also explore the possibility of performance prediction in LTE networks without the need to perform active measurements. For this, we look at the relationship between the measured signal quality and the achievable data rates and latencies. We find that while there is a strong correlation between some of the signal quality indicators and the achievable data rates, the relationship between them is stochastic, i.e., a higher signal quality makes better performance more probable but does not guarantee it. We then use our empirical measurement results as a basis for a model that uses signal quality measurements to predict a throughput distribution. The resulting estimate of the obtainable throughput can then be used in adjustable bitrate applications like video streaming to improve their dependability.

Mobile networks also task TCP congestion control algorithms with a new challenge: Usually, senders use TCP congestion control to avoid causing congestion in the network by sending too many packets and so that the network bandwidth is divided fairly. This can be a challenging task since it is not known how many senders are in the network, and the network load can change at any time. In mobile vehicular networks, TCP congestion control is confronted with the additional problem of a constantly changing capacity: As users change their location, the quality of the channel also changes, and the capacity of

the channel can experience drastic reductions even when the difference of location is very small. Additionally, in our measurements, we have observed that packet losses only rarely occur (and instead, packets are delayed and retransmitted), meaning that loss-based algorithms like Reno or CUBIC can be at a significant disadvantage. In this thesis, we compare several popular congestion control algorithms in both stationary and mobile scenarios. We find that many loss-based algorithms tend to cause bufferbloat and thus overly increase delays. At the same time, many delay-based algorithms tend to underestimate the network capacity and thus achieve data rates that are too low. The algorithm that performed the best in our measurements was TCP BBR, as it was able to utilize the full capacity of the channel without causing bufferbloat and also react to changes in capacity by adjusting its window. However, since TCP BBR can be unfair towards other algorithms in wired networks, its use could be problematic.

Finally, we also propose how our model for data rate prediction can be used to improve the dependability of mobile video streaming. For this, we develop an algorithm for adaptive bitrate streaming that provides a guarantee that the video freeze probability does not exceed a certain pre-selected upper threshold. For the algorithm to work, it needs to know the distribution of obtainable throughput. We use a simulation to verify the function of this algorithm using a distribution obtained through the previously proposed data rate prediction algorithm. In our simulation, the algorithm limited the video freeze probability as intended. However, it did so at the cost of frequent switches of video bitrate, which can diminish the quality of user experience. In future work, we want to explore the possibility of different algorithms that offer a trade-off between the video freeze probability and the frequency of bitrate switches.

ZUSAMMENFASSUNG

Die Verbesserung der Zuverlässigkeit von mobilen Netzwerk-basierten Anwendungen ist aus vielen Gründen eine komplizierte Aufgabe: Vor allem in Deutschland war die Entwicklung der Mobilfunkinfrastruktur nicht immer schnell genug, um mit der wachsenden Nachfrage Schritt zu halten. Es gibt immer noch viele Funklöchern, die für Kommunikationsausfälle verantwortlich sind. Aber auch an Orten, an denen Infrastruktur ausreichend vorhanden ist, stellt die Mobilität der Nutzer eine große Herausforderung für die Zuverlässigkeit der Anwendungen dar: Wenn sich der Nutzer bewegt, kann sich die Kapazität des Kanals stark verändern. Dies kann dazu führen, dass Anwendungen wie Videostreaming mit einstellbarer Bitrate die in der Vergangenheit erreichten Downloadraten nicht zur Vorhersage der zukünftigen Leistung nutzen können, da diese nur alte Informationen über die Datenraten an einem anderen Standort enthalten.

In dieser Arbeit untersuchen wir die Nutzung von 4G LTE für zuverlässige Kommunikation in mobilen Fahrzeugszenarien. Zu diesem Zweck untersuchen wir zunächst die Leistung von LTE, insbesondere in mobilen Umgebungen, und wie sie sich im Laufe der Zeit entwickelt hat. Wir vergleichen Messungen, die in einem zeitlichen Abstand von mehreren Jahren durchgeführt wurden, und untersuchen Leistungsunterschiede in städtischen und ländlichen Gebieten. Wir stellen fest, dass die kontinuierliche Weiterentwicklung des 4G-Standards zwar theoretisch eine bessere Leistung ermöglicht hat, dass sich dies aber aufgrund des langsamen Ausbaus der Infrastruktur, insbesondere entlang von Autobahnen, nicht immer in der Praxis bemerkbar gemacht hat.

Wir untersuchen auch die Möglichkeit der Leistungsvorhersage in LTE-Netzen, ohne aktive Messungen durchführen zu müssen. Zu diesem Zweck untersuchen wir die Beziehung zwischen der gemessenen Signalqualität und den erreichbaren Datenraten und Latenzzeiten. Wir stellen fest, dass es zwar eine starke Korrelation zwischen einigen der Signalqualitätsindikatoren und den erreichbaren Datenraten gibt, die Beziehung zwischen ihnen aber stochastisch ist, d. h. eine höhere Signalqualität macht eine bessere Leistung zwar wahrscheinlicher, garantiert sie aber nicht. Wir verwenden dann unsere empirischen Messergebnisse als Grundlage für ein Modell, das die Signalqualitätsmessungen zur Vorhersage einer Durchsatzverteilung nutzt. Die sich daraus ergebende Schätzung des erzielbaren Durchsatzes kann dann in Anwendungen mit einstellbarer Bitrate wie Videostreaming verwendet werden, um deren Zuverlässigkeit zu verbessern.

Mobile Netze stellen auch TCP Congestion Control Algorithmen vor eine neue Herausforderung: Normalerweise verwenden Sender

TCP Congestion Control, um eine Überlastung des Netzes durch das Senden von zu vielen Paketen zu vermeiden, und um die Bandbreite des Netzes gerecht aufzuteilen. Dies kann eine schwierige Aufgabe sein, da es nicht bekannt ist, wie viele Sender sich im Netz befinden, und sich die Netzlast jederzeit ändern kann. In mobilen Fahrzeugnetzen ist TCP Congestion Control mit dem zusätzlichen Problem einer sich ständig ändernden Kapazität konfrontiert: Wenn die Benutzer ihren Standort wechseln, ändert sich auch die Qualität des Kanals, und die Kanalkapazität des Kanals kann drastisch sinken, selbst wenn der Unterschied zwischen den Standorten sehr gering ist. Darüber hinaus haben wir bei unseren Messungen festgestellt, dass Paketverluste nur selten auftreten (stattdessen werden Pakete verzögert und erneut übertragen), was bedeutet, dass verlustbasierte Algorithmen wie Reno oder CUBIC einen großen Nachteil haben können. In dieser Arbeit vergleichen wir mehrere gängige Congestion Control Algorithmen sowohl in stationären als auch in mobilen Szenarien. Wir stellen fest, dass viele verlustbasierte Algorithmen dazu neigen, einen Pufferüberlauf zu verursachen und somit die Latenzen übermäßig erhöhen, während viele latenzbasierte Algorithmen dazu neigen, die Kanalkapazität zu unterschätzen und somit zu niedrige Datenraten erzielen. Der Algorithmus, der bei unseren Messungen am besten abgeschnitten hat, war TCP BBR, da er in der Lage war, die volle Kapazität des Kanals auszunutzen, ohne den Pufferfüllstand übermäßig zu erhöhen. Ebenso hat TCP BBR schnell auf Kapazitätsänderungen reagiert, indem er seine Fenstergröße angepasst hat. Da TCP BBR jedoch in kabelgebundenen Netzen gegenüber anderen Algorithmen unfair sein kann, könnte seine Verwendung problematisch sein.

Schließlich schlagen wir auch vor, wie unser Modell zur Vorhersage von Datenraten verwendet werden kann, um die Zuverlässigkeit des mobilen Videostreaming zu verbessern. Dazu entwickeln wir einen Algorithmus für Streaming mit adaptiver Bitrate, der garantiert, dass die Wahrscheinlichkeit des Anhaltens eines Videos eine bestimmte, vorher festgelegte Obergrenze nicht überschreitet. Damit der Algorithmus funktionieren kann, muss er die Verteilung des erreichbaren Durchsatzes kennen. Wir verwenden eine Simulation, um die Funktion dieses Algorithmus zu überprüfen. Hierzu verwenden wir eine Verteilung, die wir durch den zuvor vorgeschlagenen Algorithmus zur Vorhersage von Datenraten erhalten haben. In unserer Simulation begrenzte der Algorithmus die Wahrscheinlichkeit des Anhaltens von Videos wie beabsichtigt, allerdings um den Preis eines häufigen Wechsels der Videobitrate, was die Qualität der Benutzererfahrung beeinträchtigen kann. In zukünftigen Arbeiten wollen wir die Möglichkeit verschiedener Algorithmen untersuchen, die einen Kompromiss zwischen der Wahrscheinlichkeit des Anhaltens des Videos und der Häufigkeit der Bitratenwechsel bieten.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS AND ACRONYMS

3GPP	3rd Generation Partnership Project
5GC	5G Core
ABR	Adaptive Bitrate
ACK	Acknowledgement
AIMD	Additive Increase, Multiplicative Decrease
BBR	Bottleneck Bandwidth and Round-trip propagation time
BOLA	Buffer Occupancy based Lyapunov Algorithm
CA	Carrier Aggregation
CBR	Constant Bit Rate
CCDF	Complementary Cumulative Distribution Function
CDF	Cumulative Distribution Function
CDMA2000	Code Division Multiple Access 2000
CQI	Channel Quality Indicator
CRC	Cyclic Redundancy Check
CTCP	Compound Transmission Control Protocol (TCP)
Cat	Category
cwnd	congestion window
DRX	Discontinuous Reception
E-UTRAN	Evolved Universal terrestrial radio-access network
ECN	Explicit Congestion Notification
EDGE	Enhanced Data Rates for GSM Evolution
ELASTIC	fEedback Linearization Adaptive STreamIng Controller
eNodeB	evolved Node B
EPC	Evolved Packet Core
FCFS	First-Come First-Serve
FDD	Frequency division duplex

GPRS	General Packet Radio Service
GPS	Global Positioning System
GSM	Global System for Mobile Communications
HARQ	Hybrid Automatic Repeat reQuest
HLS	HTTP Live Streaming
HSPA	High Speed Packet Access
HSS	Home Subscriber Server
HTTP	Hypertext Transfer Protocol
HyStart	Hybrid Start
iid	independent and identically distributed
IP	Internet Protocol
LTE	Long Term Evolution
MCS	Modulation and Coding Scheme
MGF	Moment-Generating Function
MIMO	Multiple Input Multiple Output
MME	Mobility Management Entity
MPEG-DASH	Moving Picture Experts Group - Dynamic Adaptive Streaming over HTTP
MSS	Maximum Segment Size
NR	New Radio
OSI	Open Systems Interconnection
OWD	One-Way Delay
P-GW	Packet Data Network Gateway
PANDA	Probe AND Adaptation
PCell	Primary Cell
PDCCH	Physical Downlink Control Channel
PDSCH	Physical Downlink Shared Channel
PDU	Protocol Data Unit
PRACH	Physical Random Access Channel

PUCCH	Physical Uplink Control Channel
PUSCH	Physical Uplink Shared Channel
Q-Q	Quantile-Quantile
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase-Shift Keying
RAN	Radio Access Network
RB	Resource Block
RLC	Radio Link Control
RSRP	Reference Signal Received Power
RSRQ	Reference Signal Received Quality
RSSI	Received Signal Strength Indicator
RTT	Round Trip Time
S-GW	Serving Gateway
SCTP	Stream Control Transmission Protocol
SINR	Signal to Interference plus Noise Ratio
SMS	Short Message Service
TB	Transport Block
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UE	User Equipment
UMTS	Universal Mobile Telecommunications System
Uu	UTRAN to user
WiMAX	Worldwide Interoperability for Microwave Access

Part I

DISSERTATION

1 INTRODUCTION

As the data rates and latencies of mobile communication improve, more and more users start using the internet on their mobile devices. As of the second quarter of 2022, mobile traffic constitutes more than half (58.99%) of all website traffic worldwide [1] and is expected to grow even further. To meet the growing demands of an increasing number of users, 3rd Generation Partnership Project (3GPP), an organization responsible for the development of technical specifications for mobile telecommunications, continues to release new versions of its mobile communication specifications that improve performance and dependability.

3GPP publishes its technical specifications in so called *Releases*. The initial Phase 1 release was published in 1992 by its predecessor organization and was the first introduction of Global System for Mobile Communications (GSM). Throughout the years, further releases have defined technical specifications for General Packet Radio Service (GPRS), Enhanced Data Rates for GSM Evolution (EDGE), Universal Mobile Telecommunications System (UMTS), and High Speed Packet Access (HSPA). The development of 4G Long Term Evolution (LTE), the standard that will be the main focus of this thesis, started with the 3GPP Release 8, published in 2008. The initial rollout phase of 4G LTE started in 2010, at which time the maximum download speed was up to 10 Mbit/s. As the development of the standard continued, the maximum achievable data rates also became higher due to improvements like higher order modulation schemes, the use of higher order Multiple Input Multiple Output (MIMO), and the introduction of carrier aggregation to utilize the fragmented spectrum better.

Currently, the rollout of the features defined in 3GPP Release 15 is ongoing in Germany and many other countries. This release defines 4G LTE enhancements and introduces 5G New Radio (NR). With Release 15, LTE is supposed to provide up to ~3.5 Gbit/s in the downlink, whereas NR is supposed to be able to provide up to 10 Gbit/s.

With the number of users growing and more bandwidth becoming available, it becomes more and more necessary to rethink various mechanisms that are prevalent in modern networks. When designing network protocols, it is common to think of them as belonging to a network layer in either the Open Systems Interconnection (OSI) or the TCP/IP model. Such protocols can then be developed independently: For example, an application layer protocol like Hypertext Transfer Protocol (HTTP) does not rely on a particular physical layer implementation or a specific TCP congestion control algorithm.

The possibility to design network protocols independently from each other has made developing network applications easier; however, there are also disadvantages to this approach: Applications and algorithms often have to deal with network outages or otherwise deal with abnormal network behavior. Dealing with such events often involves making assumptions about their cause and the behavior of the network. For example, TCP congestion control algorithms often assume that packet losses are an indicator and a necessary consequence of network congestion, even though this is not necessarily the case in mobile networks. Dealing with such events can involve significant latencies. While such latencies can still be tolerated in applications like web browsing, in live or on-demand streaming applications they can significantly impact the users' Quality of Experience.

Knowing specifics about the underlying network can also help improve application performance and dependability. While Adaptive Bitrate (ABR) algorithms like MPEG-DASH are already used to adjust the application performance when network conditions change, they often assume that past network performance can help predict future network performance, which is not necessarily the case if the user equipment changes its position.

All of these considerations show that existing algorithms often already make assumptions about the underlying network; however, these assumptions are not always valid for all types of networks. In this thesis, we will investigate how inapplicable assumptions made by existing algorithms can impair the overall network performance.

We also investigate how the dependability of mobile network applications can be improved by making more applicable predictions. For this, we (i) analyze the behavior of mobile networks, (ii) use our analysis to make predictions about network behavior without having to perform network measurements and (iii) present how such predictions can be used in ABR algorithms to improve the dependability of their performance.

1.1 THESIS OUTLINE

In Chapter 2, we provide an overview of background topics for this thesis. First, we discuss the characteristics and architecture of 4G LTE. We especially focus on details relevant to mobile scenarios, where the location of the user constantly changes, and the quality of the channel between the user equipment and the base station may therefore change as well, or the user could even be out of reach for one base station and switch to a different one. We then talk about the challenges of performing network measurements in mobile networks.

In Chapter 3, we present our analysis of the behavior of mobile networks. We show that their behavior can be very volatile and vary even if the user equipment is stationary. Such changes heavily depend

on network capacity, which can change depending on the time of the day or the day of the week. When the user equipment is mobile, the capacity available to a user is even more volatile due to the changes in signal quality.

We also look at how the performance of the 4G network has developed over time. We have performed our initial measurements using a Release 8/Category (Cat) 3 modem. As new LTE Releases have been introduced, the network performance has also drastically improved through the use of carrier aggregation, which helped utilize the fragmented LTE spectrum and significantly increase the achievable data rates, as well as higher order modulation and coding schemes.

We then investigate how the choice of transport protocol affects the performance of network applications. We look at the performance of TCP and User Datagram Protocol (UDP). We also compare the performance of various congestion control algorithms and analyze how well they work in mobile networks. We especially focus on the assumptions made about the network when designing these protocols and verify how well these hold up when the algorithms are used in a mobile network. We find that loss-based algorithms can perform poorly in mobile networks because packet losses rarely occurred in our measurements, leading to very high delays due to bufferbloat. Delay-based algorithms had the opposite problem, where they underestimated network capacity.

Since it is not feasible to perform active network measurements whenever network conditions are expected to change, we analyze the relationship between the network performance and the signal quality measured by the user equipment. We find that the signal quality can be used to predict network performance; however, the relationship between the achievable throughput and signal quality is only stochastic, i.e., only the distribution of network throughput can be predicted.

Based on our analysis of the relationship between the signal quality and network performance, we present a mathematical model that estimates the distribution of the throughput based on the signal quality indicators like Reference Signal Received Power (RSRP), Reference Signal Received Quality (RSRQ), Signal to Interference plus Noise Ratio (SINR), Received Signal Strength Indicator (RSSI), and Channel Quality Indicator (CQI). This model can be found in Chapter 4.

As already discussed, having a model of the network behavior can help to improve the performance of network applications. In Chapter 5, we present an algorithm for ABR streaming applications that can be used to limit the probability of a buffer underflow (which would result in an application freeze) to a desired upper bound. This algorithm requires the knowledge of the distribution of the achievable data rates, which can be obtained as discussed in Chapter 4.

2 BACKGROUND

This thesis deals with how mobile communication can be made more dependable, focusing on high mobility use cases like vehicular communication. In this Chapter, we discuss the basics needed to analyze the dependability of mobile communication.

First, in Section 2.1, we discuss how 4G LTE works and what allows it to be used in mobile scenarios. We discuss the 4G architecture, including the backbone network, the air interface, and the scheduler. We also briefly talk about the 4G deployment in Germany.

In Section 2.2, we talk about network measurements to analyze the performance of mobile networks. Since data in LTE is divided into transport blocks and the gaps between them can vary based on system load, commonly used methods for bandwidth estimation based on packet gaps or packet trains can fail in mobile networks if their parameters are not chosen carefully. In this Section, we discuss how the choice of measurement parameters affects the measurement result and propose parameters that can be used for accurate bandwidth estimation results.

2.1 4G IN MOBILE SCENARIOS

In order to enable communication in mobile scenarios, wireless communication is required. 4G LTE is currently one of the most widely used types of wireless communication for many reasons: The 4G infrastructure is widely available, the range of communication is high, and the quality of service has been continuously improving. With Release 15, LTE is supposed to provide up to 3.5 Gbit/s of throughput in the downlink direction.

There have been efforts to develop alternatives like municipal Wi-Fi, which have been adopted successfully in some cities [2]. However, they are often only sparsely available and can suffer from performance issues [3].

In this Section, we discuss how 4G works and what makes it a good choice for mobile communication scenarios. We discuss some implementation details of the 4G network, including the core network and the details of the wireless communication.

2.1.1 *Evolved Packet Core*

In this Section, we describe the operation of the core network of 4G LTE - the Evolved Packet Core (EPC). EPC has been introduced as the

packet core of LTE in 3GPP Release 8 and is currently used as the core network for 4G as well as 5G. EPC will be replaced by the 5G Core (5GC) as soon as its deployment is finished.

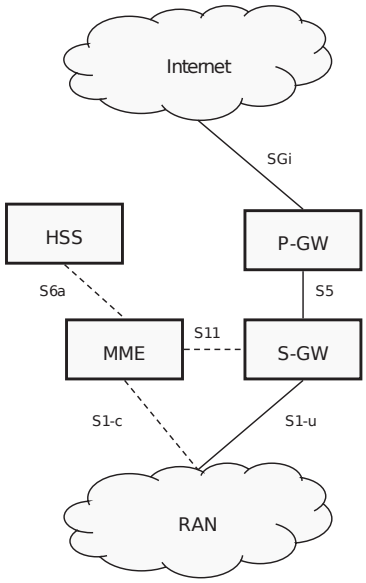


Figure 2.1: Overview of the EPC architecture [4].

Unlike the core networks of the previous cellular communication generations, the EPC no longer has a circuit-switched domain, and both data packets and voice/Short Message Service (SMS) messages are transferred via Internet Protocol (IP) packets.

In Fig. 2.1, we present an overview of the EPC structure. The main components of the EPC are Mobility Management Entity (MME), Serving Gateway (S-GW), Packet Data Network Gateway (P-GW) and Home Subscriber Server (HSS):

The MME is the control-plane node of the EPC. It is responsible for control-plane operations like connection establishment or release, change of the activity state of the User Equipment (UE), handover signaling, handling of security keys, etc. The interface between the evolved Node B (eNodeB) and the MME is called the S1-c interface. This communication is done using the Stream Control Transmission Protocol (SCTP).

The S-GW is the user-plane node that connects the EPC to the Evolved Universal terrestrial radio-access network (E-UTRAN). It routes and forwards packets from the PDN-GW to the eNodeB and handles inter-eNodeB handovers. The interface between the eNodeB and the S-GW is called the S1-u interface. It handles data delivery of Protocol Data Units (PDUs) between the eNodeB and the S-GW.

The P-GW connects the EPC to external packet data networks, most notably the Internet. For this, the P-GW handles IP address assignment to UE. It also connects the EPC to non-3GPP radio-access technologies like Code Division Multiple Access 2000 (CDMA2000) or Worldwide Interoperability for Microwave Access (WiMAX). The P-GW communicates with the S-GW via the S5 interface and the Internet via the SGi interface.

The HSS is a central database that contains user-related and subscription-related information. It performs mobility management, call and session establishment support, user authentication, and access authorization.

2.1.2 *Air Interface*

The wireless communication between the UE and the EPC is handled by the E-UTRAN.

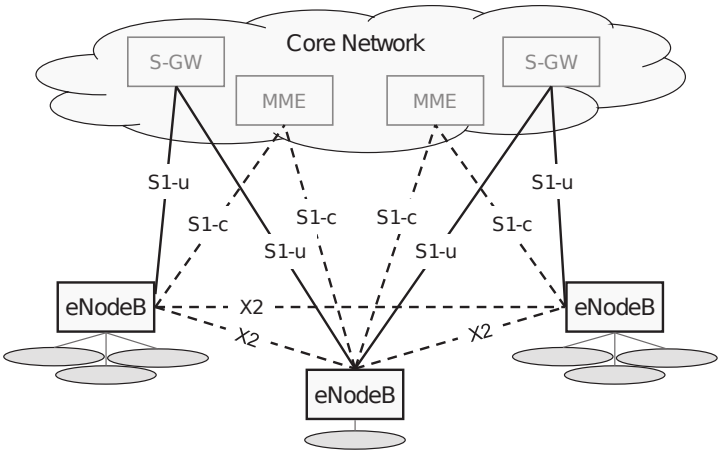


Figure 2.2: Overview of the Radio Access Network (RAN) architecture [4].

The E-UTRAN only has a single type of node - the eNodeB. The eNodeB is connected to the EPC via the S1 interface. Specifically, it is connected to the MME via the S1-c interface for control-plane communication and to the S-GW via the S1-u interface for user-plane communication. eNodeBs are connected to each other via the X2 interface to support active-mode mobility and packet forwarding during handovers. An overview of the overall architecture can be seen in Fig. 2.2

The interface between the eNodeB and the UE is called the UTRAN to user (Uu) interface. Here, the UE and the eNodeB exchange information via a wireless link.

2.1.2.1 *LTE Spectrum*

The channel between the eNodeB and the UE can be located in different frequency bands and have different bandwidths. For example, in Germany, the 800 MHz, 1800 MHz, and 2600 MHz frequency bands can

be used for LTE communication. Different providers license differently-sized frequency ranges in these bands and can divide these into differently-sized channels. In LTE the channel size can be 1.4 MHz, 3 MHz, 5 MHz, 10 MHz, 15 MHz or 20 MHz. In most European countries, Frequency division duplex (FDD) is used to separate downlink and uplink communication, which means that the downlink and uplink channels are located in two different frequency ranges. An overview of LTE bands commonly used in Germany and their respective frequency ranges can be found in Table 2.1.

Band	Range	Uplink	Downlink	Channel Bandwidth
3	1800 MHz	1710-1785 MHz	1805-1880 MHz	1.4, 3, 5, 10, 15, 20 MHz
7	2600 MHz	2500-2570 MHz	2620-2690 MHz	5, 10, 15, 20 MHz
8	900 MHz	880-915 MHz	925-960 MHz	1.4, 3, 5, 10 MHz
20	800 MHz	832-862 MHz	791-821 MHz	5, 10, 15, 20 MHz

Table 2.1: Overview of LTE bands commonly used in Germany and their respective frequency ranges.

The LTE spectrum is divided into so-called subcarriers, which can be either 7.5 kHz or 15 kHz wide. These subcarriers are organized into Resource Blocks (RBs), which consist of either 12x15 kHz or 24x7.5 kHz subcarriers, i.e., they have a total bandwidth of 180 kHz. These RBs have a duration of 0.5 ms, also called a slot. Multiple RBs can then be used to form differently-sized channels. An overview of how many RBs are needed for channels with a different bandwidth can be found in Table 2.2.

Bandwidth	RBs
1.4 MHz	6
3 MHz	15
5 MHz	25
10 MHz	50
15 MHz	75
20 MHz	100

Table 2.2: RBs needed for channels with different bandwidth in LTE. Note that 10% of the bandwidth is used for a Guard Band (except 1.4 MHz).

Two RBs constitute a subframe (1 ms), which is the smallest unit of resources that can be assigned to a user by a scheduler. Each subframe carries a Transport Block (TB), which is the smallest data unit that can be scheduled for a user, and a Cyclic Redundancy Check (CRC) checksum. Further, the subframes are organized in frames with a duration of 10 ms, i.e., 10 subframes.

One problem of LTE is that the standard only defines channels of up to 20 MHz, limiting the maximum possible transmission rate. Carrier aggregation was introduced to the LTE standard in Release 12 [5] to alleviate this problem. It allows the combined use of multiple component carriers with up to 20 MHz each, which can be used to increase the maximum transmission rate or load balancing. We further discuss carrier aggregation in Section 3.3.

2.1.2.2 *LTE Channels*

LTE has multiple different channels to transmit different types of information. An Overview of downlink channels can be found in Fig. 2.3, for uplink channels it can be seen in Fig. 2.4:

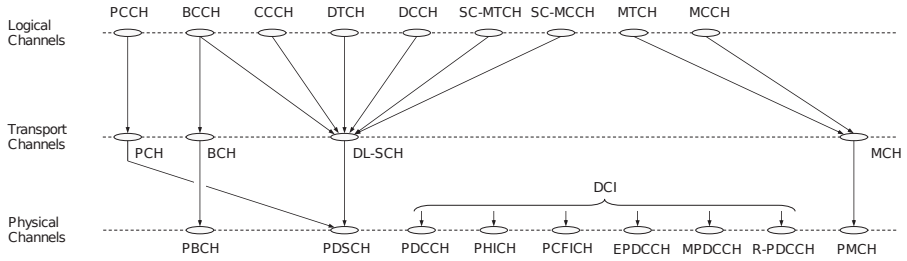


Figure 2.3: LTE downlink channels [4].

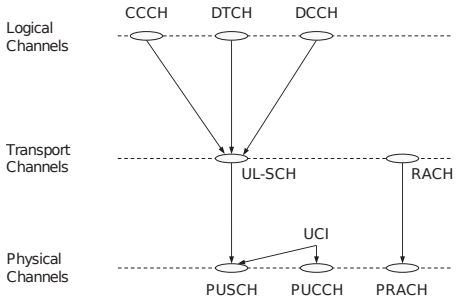


Figure 2.4: LTE uplink channels [4].

In this Section, we are going to briefly explain the function of some of the channels, i.e., Physical Random Access Channel (PRACH), Physical Downlink Shared Channel (PDSCH), Physical Downlink Control Channel (PDCCH), Physical Uplink Shared Channel (PUSCH), and Physical Uplink Control Channel (PUCCH):

PRACH is used by the UE to establish the initial connection. For this, the UE transmits a random access preamble, which the eNodeB uses to estimate the delay to adjust uplink timing.

PDSCH and PUSCH are the physical channels that carry the data of all users. The data is divided into RBs as described in Section 2.1.2.1.

PDCCH and PUCCH are the control channels that carry the scheduling assignments for the data that is transported on the PDSCH and PUSCH channels. They indicate to the UEs which RBs belong to them. The UEs need to periodically scan these channels to find new scheduling assignments. Since this can consume a lot of power, a power-saving mechanism called Discontinuous Reception (DRX) has been introduced to alleviate this problem. We discuss its detail in the next Section.

2.1.3 *DRX*

To be able to send or receive data on the PDSCH/PUSCH channels, a UE needs to scan the PDCCH/PUCCH control channels for scheduling assignments. When it comes to the PDCCH, the UE doesn't know in advance when it will receive new data, so it needs to monitor the downlink control channel continuously. This process can consume a lot of power, which is why LTE UEs implement a power-saving mechanism called DRX. The main idea behind DRX is for the UE to scan the control channel less often when no data has been sent or received for some specified period of time. This is achieved by introducing cycles of different lengths, during which the UE scans the control channel with different frequencies. The UE changes the cycle duration when no packet has been received for some time. For this, the UE uses an inactivity timer that is restarted every time the UE is scheduled.

A UE can implement a long and a short DRX cycle to handle different usage scenarios: A short DRX cycle that is meant to handle applications with periodic, but not continuous activity (e.g., voice over IP), and a long DRX cycle to handle applications that have long periods of silence (e.g., a user interacting with a website). In [6], the authors have analyzed a live LTE network and found that an UE had switched from continuous reception to a short DRX cycle when the UE was inactive between 200 ms and 2.5 s. For inactivity times between 2.5 s and 10.5 s, the UE enters a long DRX cycle. When the inactivity exceeds 10.5 s, the UE becomes idle.

2.1.4 *LTE Deployment*

Since the start of the LTE rollout in 2010, its deployment has continuously advanced. As of the end of 2021, the three most significant German LTE providers, Telekom, Vodafone, and O2, claim that they each achieve coverage of more than 98% of the total population of Germany. In practice, users often report coverage gaps, especially in rural areas. We discuss our experiences with coverage gaps in Chapter 3.

A single LTE cell can have a size anywhere between a few tens of meters and up to 100 km depending on the utilized frequency and signal power. In order to cover larger areas, multiple cells have to be

used. To reduce interference between cells that use the same frequency, they should ideally cover non-overlapping areas. This way of arranging cells is also called space division multiplexing. The area covered by a cell is usually best represented by a circle around the eNodeB in the middle of the cell. For a more convenient visual representation, the coverage of a cell is usually modeled as a hexagon, which makes it easier to arrange multiple cells in a grid than it would be with circles. An example of such an arrangement can be seen in Fig. 2.5:

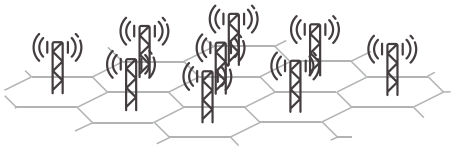


Figure 2.5: Example of an arrangement of LTE cells in a hexagonal grid [4].

Not all users in the same cell receive an equally good service: The further away a user is from the middle of the cell, i.e., the eNodeB, the more they will experience a reduction of signal quality due to attenuation. When the signal quality is reduced, the Modulation and Coding Scheme (MCS) is also changed so that it is still possible to decode the signal in the presence of noise. Depending on the strength of the signal, LTE uses either Quadrature Phase-Shift Keying (QPSK) or different constellations of Quadrature Amplitude Modulation (QAM). An overview of which MCS is used for which signal quality can be seen in Table 2.3.

The CQI is a value determined by the UE that indicates to the eNodeB what the highest modulation and code rate are with which the UE can receive a single TB with an error probability not exceeding 0.1 [7]. The UE periodically reports this value to the eNodeB so that an appropriate MCS can be selected. When the CQI value is low, an MCS with fewer bits per symbol and a higher redundancy is selected so that it is easier to distinguish the weaker signal from noise.

When the UE moves even further away from the middle of a cell, it is possible that no MCS can be selected that can keep the packet loss under a certain threshold. When this happens, it becomes necessary to change the cell. Usually, there is a predefined signal quality threshold under which a cell handover becomes necessary. This threshold exists to prevent the UE from being handed back and forth between two cells.

A handover may occur during an ongoing data transmission, i.e., the eNodeB still has some packets to transmit to the UE. In such a case, the eNodeB would use the X2 interface to forward the data packets to the new eNodeB so that the UE can still receive the packets. This kind of seamless handover is also one of the reasons why LTE works well in mobile scenarios.

LTE can also be used as a technology enabling mobile Internet in public transportation even to users without a subscription: Frequently,

CQI index	modulation	code rate x 1024	efficiency
0	out of range		
1	QPSK	78	0.1523
2	QPSK	193	0.3770
3	QPSK	449	0.8770
4	16-QAM	378	1.4766
5	16-QAM	490	1.9141
6	16-QAM	616	2.4063
7	64-QAM	466	2.7305
8	64-QAM	567	3.3223
9	64-QAM	666	3.9023
10	64-QAM	772	4.5234
11	64-QAM	873	5.1152
12	256-QAM	711	5.5547
13	256-QAM	797	6.2266
14	256-QAM	885	6.9141
15	256-QAM	948	7.4063

Table 2.3: CQI to modulation scheme mapping in LTE Release 12 [5].

trains or buses have LTE routers installed that allow users to connect to them via Wi-Fi and use a limited amount of data for free.

2.1.5 *LTE Scheduler*

In an LTE cell, the cell's capacity is shared among all users. In order to divide the available resources between the users, the eNodeB uses a scheduler to assign time or frequency resources for the data of different users.

The implementation of LTE schedulers is not standardized and is left up to the network providers. However, there are several common strategies that are typically used:

MAX-C/I (CARRIER-TO-INTERFERENCE) OR MAXIMUM RATE This type of scheduler tries to optimize the utilization of available resources by assigning the most scheduling slots to the users with better channel conditions. It achieves the least fairness as it will starve users with bad channel quality.

ROUND-ROBIN This type of scheduler serves all users in alternate order, which maximizes the fairness between users. However, this can lead to poor resource utilization.

PROPORTIONAL FAIR This type of scheduler tries to improve the fairness of max-C/I schedulers by considering the individual users' instantaneous channel quality and scheduling users with the best relative channel conditions. Since relative channel conditions can often change due to fast fading, even users with bad channel quality can be scheduled when their relative channel conditions improve.

Overall, the choice of the scheduling strategy is always a trade-off between fairness and efficient resource utilization: When trying to optimize the use of resources (as it is done in max-C/I schedulers), some users with bad channel quality may not get any service at all. On the other hand, a round-robin scheduler that tries to optimize fairness may waste resources. The proportional fair scheduling strategy tries to strike a balance between the two.

For any given eNodeB, it is unknown which scheduling strategy it implements. Different network providers might use different strategies based on location or the type of traffic. The open-source srsENB project allows one to choose between proportional-fair and round-robin schedulers [8].

2.2 AVAILABLE BANDWIDTH ESTIMATION IN MOBILE NETWORKS

For mobile applications to operate efficiently, it can be helpful for them to know what data rates can be achieved at a particular moment. Due to the previously discussed limitations of the mobile networks, the achievable data rates can vary drastically when the UE is mobile. Some variations can even be expected when the user is stationary.

An application can determine what data rates can be achieved in a network using *available bandwidth estimation* algorithms. Here, the term bandwidth refers to the achievable data rates and not the bandwidth of the physical channel. Many such algorithms already exist that work well for wired networks; an overview can be found in [9]. These algorithms often assume a network with First-Come First-Serve (FCFS) scheduling and a single tight link, i.e., the link with the minimum available bandwidth. This tight link has a capacity of C that is shared by multiple senders. The algorithm aims to find the remainder of the capacity not used by other senders.

This is usually done by sending probe traffic with gaps of a certain width between the packets: When the selected gap between the packet g_{in} is the same as the gap g_{out} measured at the receiver, then it means that the sending rate was equal to or lower than the available bandwidth. When g_{out} is greater than g_{in}, it means that the packets had to be queued at the tight link, either because of other senders or limited link capacity.

There are multiple categories to distinguish between different available bandwidth estimation methods: active vs. passive, packet pair vs. packet train, and iterative vs. direct:

ACTIVE VS. PASSIVE The first question when performing available bandwidth estimation is whether the algorithm creates its own traffic or uses existing traffic like, e.g., TCP traffic. When the method creates its own traffic, it is referred to as active. Active methods tend to be more precise as the sender has complete control of the traffic. However, they also affect the state of the network by creating load themselves. When existing traffic is used, the method is referred to as passive. Such methods are often less precise because a different application controls the traffic pattern.

PACKET PAIR VS. PACKET TRAIN When we use an active bandwidth estimation technique and can define our traffic pattern, the next question is what our traffic pattern should be. In the simplest case, we deal with packet pairs [10], i.e., two packets with a gap g_{in} between them. Alternatively, we can send multiple packets with the same gaps between all of them, i.e., a packet train [11, 12]. If the gaps between the packets in the packet train are not constant but increasing, we then speak of packet chirps [13]. Sending packet trains instead of packet pairs can help reduce the noise in the measurements but also increases the load on the network.

ITERATIVE VS. DIRECT The next question is how the method uses packet pairs/trains/chirps to discover the available bandwidth. Iterative methods try to find the point where the gap at the receiver starts to increase, e.g., by performing a binary search (Pathload [14]) or sending packet chirps (Pathchirp [13]). Direct methods like Spruce [15], TOPP [16] or DietTOPP [17] try to set the input gap g_{in} so that the rate exceeds the capacity of the link and then estimate the available bandwidth from the measured output gap g_{out}.

However, these methods often do not work well in some types of networks as they assume FCFS scheduling. For 802.11 networks, it has been shown that existing algorithms find the fair share, which can be very different from the available bandwidth because the assumption of FCFS is violated [18].

The use of such methods faces two major obstacles in LTE: First, the scheduler used by LTE's eNodeBs is not necessarily an FCFS scheduler. Max-C/I or proportional fair schedulers use the information of the UE's signal quality to send more data to those UEs that have a better access to the channel. Second, before sending packets to the UE, the scheduler combines the packets into TBs with a duration of 1 ms, thus reducing or increasing the gaps between the packets without it being an indication of bandwidth limitation [19]. Methods like Ookla's

Speedtest that measure the bandwidth that can be achieved by a long-lived TCP connection still work; however, they require either a long measurement time or a lot of data.

2.2.1 *Bandwidth Estimation with Packet Trains*

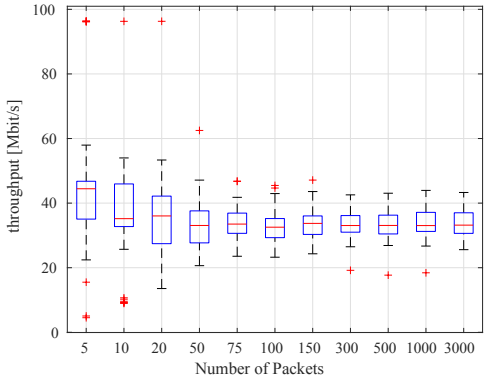


Figure 2.6: Comparison of measured throughput for packet trains of different lengths with full-sized packets. Boxplots depict 0.05, 0.25, 0.5, 0.75, and 0.95 quantiles.

As discussed in the previous Section, we do not expect packet pair based methods for available bandwidth estimation to work well in mobile networks. However, packet train based methods should still work if the packet train spans multiple transport blocks. To analyze how many packets a packet train should contain for a reliable bandwidth estimate, we perform a series of UDP Constant Bit Rate (CBR) measurements where we send $n \in \{5, 10, 20, 50, 100, 150, 300, 500, 1000, 3000\}$ packets with a size of 1400 Bytes. We use a sending rate that is higher than the observed network capacity so that the send buffer never becomes empty. For each n, we send 50 packet trains and calculate the throughput at the receiver. We remove the first packet from each packet train when calculating the throughput as we have noticed that it gets excessively delayed due to the power-saving mechanisms in LTE that we have discussed in Section 2.1.3. In Fig. 2.6, we show a boxplot with the distribution of the obtained throughput for each n. Here we can see that the estimated throughput and its variability are much higher for smaller packet trains than for larger ones. This is because of the previously discussed grouping of packets into transport blocks: smaller packet trains will often be combined into one transport block, resulting in reduced packet gaps and therefore an overestimated throughput. Packets may also be put into separate transport blocks, resulting in an increased packet gap and a very low throughput estimate. Both of these effects can be seen for packet trains with less than 50 packets.

In order to obtain more reliable estimates of the throughput, we, therefore, have to send larger packet trains. This increases the probability that multiple packets are spread over multiple transport blocks. In Fig. 2.6, we see that when there are more than 50 packets in a packet train, the estimate becomes more stable and is also, on average, the same regardless of the exact count of the packets.

Therefore, we conclude that using packet trains can be a reliable way of throughput estimation. However, the number of packets has to be large enough to avoid being affected by division into transport blocks.

2.2.2 *Bandwidth Estimation with Smaller Packets*

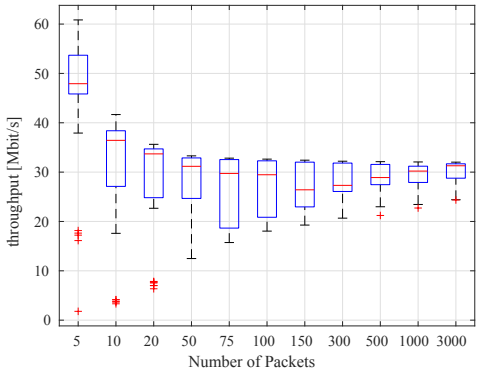


Figure 2.7: Comparison of measured throughput for packet trains of different lengths with 500 B packets. Boxplots depict 0.05, 0.25, 0.5, 0.75, and 0.95 quantiles.

Next, we wanted to see if making the packets smaller could help us reduce the data needed to obtain an accurate throughput estimate. For this, we select a packet size of 500 Bytes and repeat the experiment described above. The resulting throughput estimates can be seen in Fig. 2.7. It can be seen that with smaller packet trains, the variability of the throughput estimates becomes even more prominent for smaller packet trains. This is because the division of packets into transport blocks depends on the size of the packets and not their number. This means that now even for larger packet trains, we have some probability of either the scheduler combining them into one transport block resulting in an overestimated throughput, or dividing them into multiple transport blocks in a way that reduces the throughput estimate. The throughput estimate becomes stable only for packet trains with 300-500 packets.

We conclude that there is no advantage to using smaller packets in packet trains when performing bandwidth estimation in an LTE network, as the division into transport blocks is based on the packet

size. This means that the same amount of data must still be transmitted to obtain a reliable estimate of the available bandwidth.

3 PERFORMANCE EVALUATION

In order to be able to create a model for the prediction of the behavior of LTE, we first need to analyze the performance of the existing network.

In Section 3.1, we look at the throughput and delays that can be achieved and their dependence on different factors like frequency, bandwidth, and signal quality, as well as factors like location, time, and day.

In Section 3.2, we analyze the performance of different popular congestion control algorithms in LTE. For this, we analyze stationary scenarios with a near-constant capacity and mobile scenarios where the channel's capacity constantly changes.

In Section 3.3, we look at the performance of carrier aggregation, a technique used to combine multiple LTE component carriers for more bandwidth. We analyze the possible achievable performance improvements regarding throughput and delays, availability in our region, and behavior during handovers.

3.1 4G LTE MOBILE AND STATIONARY PERFORMANCE

In this Section, we analyze the performance of LTE. For this, we perform mobile and stationary measurement campaigns in a major commercial LTE network. In our mobile measurements, we analyze how parameters like the frequency band, the corresponding bandwidth, and the channel's signal quality affect the achievable performance and how this has changed over time. In addition, we compare the performance of the commonly used TCP and UDP transport protocols. Finally, we also look at the differences in performance depending on the time of usage. This Section is partially based on [20]. It includes additional measurement results and evaluations.

3.1.1 *Methodology & Measurement Setup*

First, we describe our measurement methodology and setup. Our measurements have been performed over several years using different hardware setups as the development of the LTE standard has progressed. For our LTE measurements, we have used the following modems:

- *Sierra Wireless AirPrime MC7304* [21], an LTE Cat 3 [22] modem with up to 100 Mbit/s downlink and 50 Mbit/s uplink throughput, installed in a *PCEngines alix6f2* system board [23].

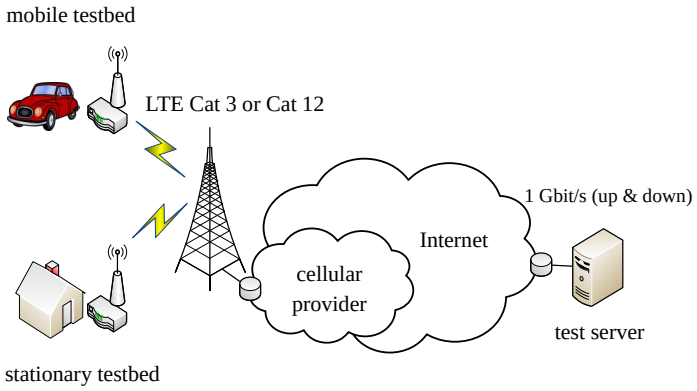


Figure 3.1: Topology of the LTE measurement setup.

- *Sierra Wireless AirPrime EM7565* [24], an LTE Cat 12 [5] modem with up to 600 Mbit/s downlink and up to 150 Mbit/s uplink throughput, installed in a *PCEngines apu3c4* system board [23].

For stationary measurements, we have used two LTE quad-band omnidirectional antennas. For mobile measurements, we have used a *Panorama Antennas LGMM-7-27-24-58* MIMO antenna for vehicular communication [25] installed in a *VW Golf VII*.

In order to send and receive data to/from the 4G modem, we have set up a test server connected to the Internet via the X-win network, i.e., the German national research network, at a rate of 1 Gbit/s.

For stationary measurements, we keep the LTE testbed at a fixed location in our institute. For mobile measurements, we repeatedly drive on a city and highway route between Hanover and Brunswick. This route is approximately 66 km and includes urban and rural Sections. The speed in the city is around 50 km/h, while on the highway it is 100-130 km/h. An overview of our measurement setup can be seen in Fig. 3.1.

When performing measurements, we record throughput and delay values, as well as other information like the cell ID, frequency, bandwidth, signal quality indicators, i.e., RSRP, RSRQ, RSSI, SINR and CQI, as well as vehicle speed and Global Positioning System (GPS) position. We also record the time of the measurement in order to analyze how LTE performance changes over the day and week.

We also investigate the performance of different transport layer protocols used by infotainment applications, since many streaming services are based on UDP, whereas TCP is commonly used for file downloads.

(a) Highway route, where measurements have been performed in 800 MHz cells with 10 MHz bandwidth.

(b) City route, where measurements have been performed in 1800 MHz cells with 20 MHz bandwidth.

Figure 3.2: Highway and city routes where measurements have been performed. Map data © OpenStreetMap contributors, CC BY-SA [26].

3.1.2 *Mobile Measurements*

In the following, we present results from mobile measurements performed on the route between Hanover and Brunswick.

We initially performed these measurements with an LTE Cat 3 modem. The results of these are published in [20]. Additionally, we have repeated these measurements with an LTE Cat 12 modem. In this Section, we evaluate and compare the results of both measurement campaigns.

3.1.2.1 *Impact of the cell frequency & bandwidth*

We have observed 13 cells in the LTE B3 band, which is located in the 1800 MHz frequency band. The median and maximum distance between these cells is 738 m and 1500 m, respectively. Furthermore, we discovered 19 cells in the LTE B20 band, which is located in the 800 MHz frequency band. Here, the cell coverage is significantly more extensive, namely 2054 m in the median and up to 12.2 km as the maximum. By investigating the bands' spatial distribution, we identify a clear trend, i.e., B3 cells in the city of Hanover and Brunswick and B20 on the highway and in rural areas. The only exception is a small intersection at the edge of the cities, where we observed both. This distribution is explained by the fact that higher frequencies, such as 1800 MHz, can only cover smaller regions. Consequently, the LTE infrastructure in rural areas often consists of only cells using the B20 band as it would not be cost-efficient to use a more dense network of cells that use the B3 band in sparsely populated areas.

In our LTE Cat 3 measurements in the B3 band, we have observed a bandwidth of 10 MHz, and for the measurements in the B20 band a bandwidth of 20 MHz. The reason is that there is a total of 30 MHz bandwidth available in the B20 band and 75 MHz in the B3 band [27]. These bands are currently licensed by 3 German providers, resulting in 10 MHz of bandwidth being available in the B20 band and at least 20 MHz in the B3 band for each of them [28].

With Release 10/Cat 6, carrier aggregation has been added to LTE [29]. Before carrier aggregation was introduced, UEs were limited to using only one component carrier per connection. Since single component

carriers are limited to a bandwidth of up to 20 MHz in LTE, the resulting throughput is also limited by that constraint. The main purpose of carrier aggregation is to increase the capacity available to users by combining multiple component carriers into a single virtual carrier. The number of component carriers that can be combined has increased from 2 in Release 10/Cat 6 to 3 in Release 12/Cat 12. Starting with Release 14/Cat 17, up to 32 component carriers can be combined [4]. To achieve this, an eNodeB capable of using carrier aggregation can use its scheduler to assign TB transmissions to multiple carriers instead of just one, thus increasing the transmission capacity. The UE has to receive the transport blocks from all component carriers and then reassemble them in the correct order before higher layers can process the data. Further details describing carrier aggregation implementation can be found in [4, 30].

The component carriers used by Carrier Aggregation (CA) can be in the same or different frequencies. For example, suppose a network provider has 30 MHz of bandwidth in a frequency band. In that case, the bandwidth in this frequency band can be split into 2 component carriers (e.g., 15 MHz each). These component carriers can be combined into a single virtual component carrier with 30 MHz of bandwidth. If the network provider has an additional component carrier in a different frequency band, it can also be combined with the other component carriers.

In our measurements with the LTE Cat 12 modem, we have observed that carrier aggregation was activated with component carriers in the bands B20 (10 MHz) and B3 (20 MHz), which we denote as B20_B3 (also known as CA_3A-20A), as well as within the B3 band, where a 10 MHz carrier and a 20 MHz carrier were combined, which we denote as B3_B3 (also known as CA_3C). We have not observed carrier aggregation with three or more carriers in any of our mobile or stationary measurements. We have also observed measurements in the B8 band with a bandwidth of 5 MHz, which was introduced on 17.03.2017 in Germany.

In Fig. 3.3, we present a boxplot of bitrates that have been achieved in cells with different bandwidths in our mobile measurements with the LTE Cat 3 modem. We observe a median throughput of 13 Mbit/s and 29 Mbit/s for 10 MHz and 20 MHz cells, respectively, showing that a higher bandwidth also results in higher achieved data rates. Note that since these are the results of our mobile measurements, the measurements have been performed in different locations and are, therefore, not directly comparable because of different channel conditions.

In Fig. 3.4, we present a boxplot of bitrates achieved in cells with different bandwidths in our mobile measurements with the Cat 12 modem. We do not differentiate between cells using single or multiple component carriers to achieve the total bandwidth. Here, we can again

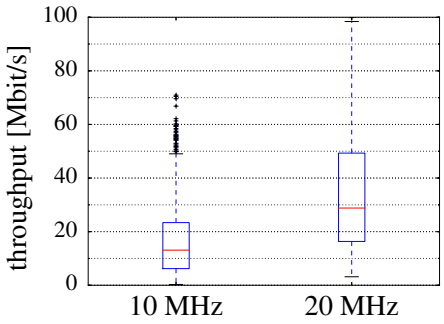


Figure 3.3: Comparison of download speeds in cells with different bandwidth for LTE Cat 3 measurements. Boxplots depict 0.05, 0.25, 0.5, 0.75, and 0.95 quantiles.

see that a higher bandwidth also usually results in a higher bitrate: The median bitrates are 7.8 Mbit/s for 5 MHz cells, 9.3 Mbit/s for 10 MHz cells, 22 Mbit/s for 20 MHz cells and 34.2 Mbit/s for 30 MHz cells. Note that even though the measurements have been performed at a later time with a modem that can achieve higher data rates according to its specification, the median data rates for the same bandwidths are lower than for the Cat 3 modem. Also note that these results are not directly comparable because of different measurement locations - the measurements were taken on the same route, but possibly not in the same spots. However, this still shows that even though the improvements in the LTE specification can enable higher data rates, these cannot always be observed when evaluating data rates in a live network.

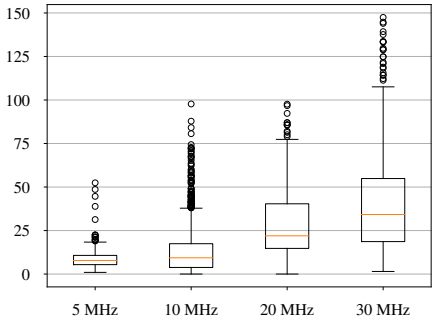
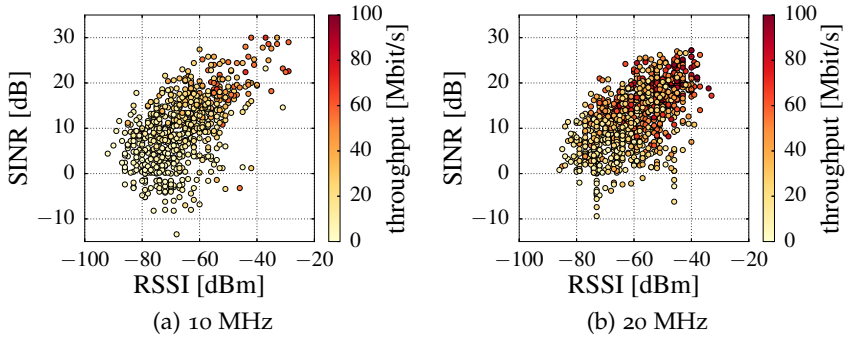


Figure 3.4: Comparison of download speeds in cells with different bandwidth for LTE Cat 12 measurements. Boxplots depict 0.05, 0.25, 0.5, 0.75, and 0.95 quantiles.

In Figs. 3.2a and 3.2b, we present maps of every performed downlink throughput measurement for the highway and city routes performed with the LTE Cat 3 modem. Each marker represents a single measurement. Red markers indicate rates of 20 Mbit/s or less, yellow markers rates between 20 and 30 Mbit/s, and green markers rates of 30 Mbit/s or more. It can be seen that the throughput generally tends to be



(a) 10 MHz (b) 20 MHz

Figure 3.5: Downlink throughput separated by RSSI and SINR for LTE Cat 3 measurements.

lower on the highway route and higher on the city route, which also corresponds to our observation that cells along the highway tend to have a lower bandwidth than the cells in the city. This has also not changed for our more recent measurements with the Cat 12 modem.

Next, we analyze the download speed dependent on bandwidth in Section 3.1.2.1. Based on these results, we show the impact of signal quality indicators SINR, RSSI, RSRP, and RSSI in Section 3.1.2.2 and investigate the spatial variability of the throughput in Section 4.2.

3.1.2.2 *Impact of the signal quality*

In the following, we present the analysis concerning the influence of signal quality parameters.

For our measurements with the LTE Cat 3 modems, we have originally only recorded RSSI and SINR values for each measurement. For the measurements with the LTE Cat 12 modem, we have additionally recorded RSRP and RSRQ values.

In Fig. 3.5 we show the download speed as a function of SINR and RSSI. We observe a general tendency of higher rates for better signal quality.

In order to evaluate the relation between SINR, RSSI, and download speed, we calculate Pearson's correlation coefficients r, see Table 3.1. For both 10 MHz and 20 MHz cells, we observe a high correlation between signal metrics and throughput (the correlation is considered high if $r > |0.5|$, see [31]).

	RSSI	SINR
Download speed 10 MHz	0.64	0.69
Download speed 20 MHz	0.53	0.61

Table 3.1: Correlation between achieved download speed and signal properties in 10 MHz and 20 MHz cells for LTE Cat 3 measurements.

In Table 3.2, we show the correlation coefficients for the LTE Cat 12 measurements which now include RSRP and RSRQ values. Overall, the correlation between the signal quality and the achieved throughput is somewhat lower than for the Cat 3 measurements, especially for the RSSI values. RSRP and RSRQ values; however, both have a higher correlation with the achieved throughput than RSSI. Only for 30 MHz bandwidth the correlation between the achieved throughput and RSRQ is surprisingly low. For both Cat 3 and Cat 12 measurements, SINR has the highest correlation with the achieved throughput.

In Fig. 3.6, we also show how the measured throughput depends on RSRP and SINR for the LTE Cat 12 measurements. Once again, it can be seen that the throughput tends to be higher when the signal quality is higher.

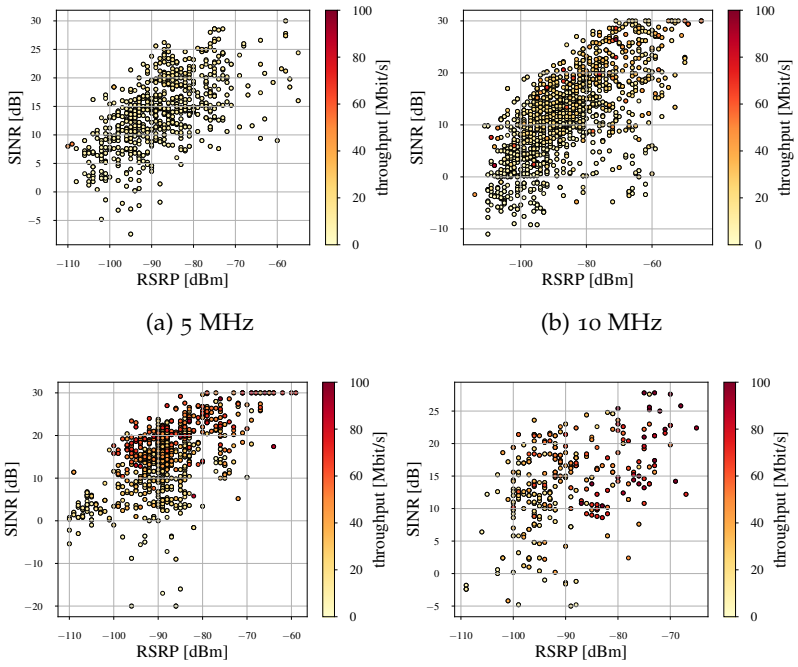
	RSSI	SINR	RSRP	RSRQ
Download speed 5 MHz	0.2	0.48	0.26	0.38
Download speed 10 MHz	0.34	0.53	0.39	0.36
Download speed 20 MHz	0.17	0.51	0.26	0.30
Download speed 30 MHz	0.56	0.44	0.68	-0.12

Table 3.2: Correlation between achieved download speed and signal properties in 5-30 MHz cells for LTE Cat 12 measurements.

When we put the signal quality values into larger groups, the relationship between the signal quality and the throughput becomes more apparent. In Fig. 3.7, we visualize this for our LTE Cat 3 measurements: When grouping SINR values (i.e., the signal quality indicator with the highest correlation with the obtained throughput) into groups of 10 dB, it becomes apparent that with a higher range of SINR-values the throughput also increases. However, even with a high signal quality, it is still possible to obtain a low throughput.

The same can be seen in Fig. 3.8 for the LTE Cat 12 measurements. Note that for lower channel bandwidth values like 5 MHz, the differences in obtainable throughput depending on signal quality are not very high. This is because the channel's capacity is smaller for lower channel bandwidths, and not much improvement can be achieved by having a higher signal quality.

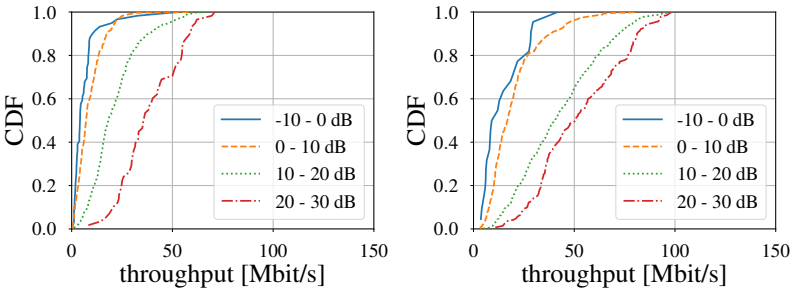
When comparing throughput results for the same bandwidth and signal quality between LTE Cat 3 and LTE Cat 12 modems, it once again becomes apparent that the modem with higher capabilities does not necessarily achieve higher throughput. Since the measurements have been performed at different times and not necessarily at the exact same locations, this effect could be caused by different cell loads. It is also possible that the provider's infrastructure has not supported the higher capabilities of the Cat 12 modem at the time of the measurements.



(a) 5 MHz (b) 10 MHz

(c) 20 MHz (d) 30 MHz

Figure 3.6: Downlink throughput separated by RSRP and SINR for LTE Cat 12 measurements.



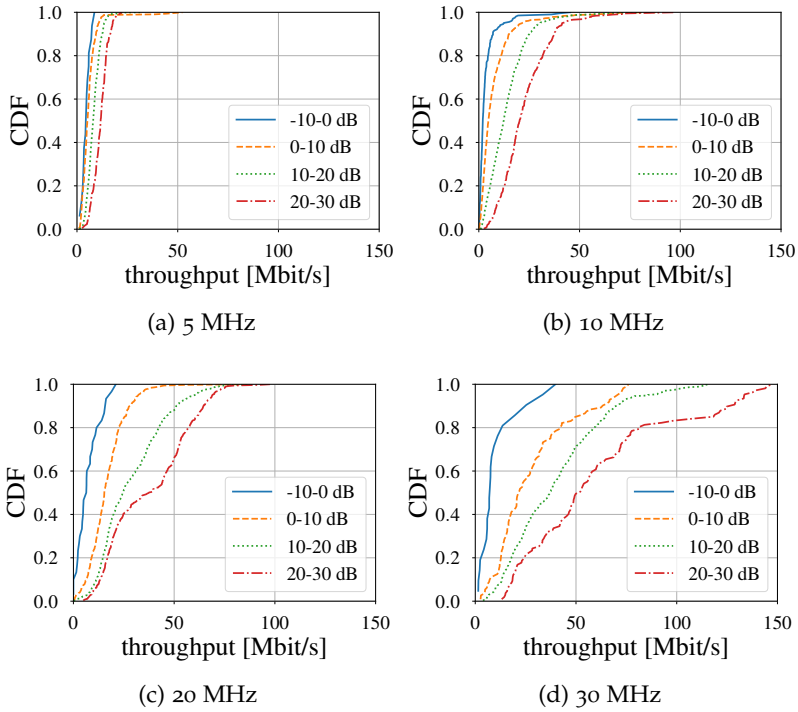
(a) 10 MHz (b) 20 MHz

Figure 3.7: Downlink throughput separated by SINR values in 10 dB groups for LTE Cat 3 measurements.

3.1.3 *Stationary Measurements*

In the previous Section, we investigated LTE's downlink performance in mobile scenarios. We observed high variations in the achievable throughput, which we have attributed to differences in signal quality, different cells, and different performance and load at different locations.

Implicitly, we assumed a stationary throughput performance over time, i.e., no differences between, e.g., 6 a.m. and 6 p.m. Investigations in [32] have shown time-of-day-dependent performance differences in



(a) 5 MHz (b) 10 MHz

(c) 20 MHz (d) 30 MHz

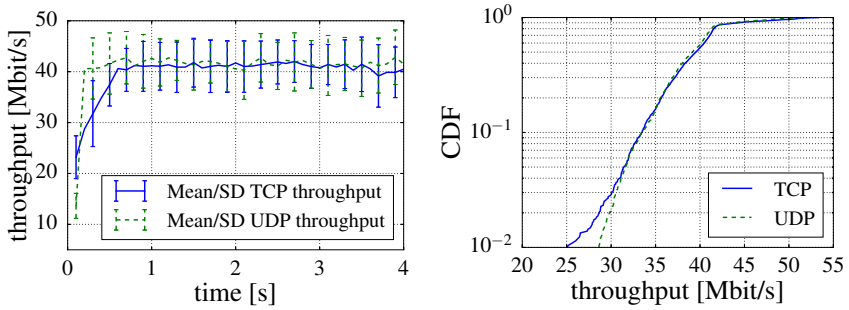
Figure 3.8: Downlink throughput separated by SINR values in 10 dB groups for LTE Cat 12 measurements.

cellular networks, where the throughput improves at night. This effect is explained by a lower network load at night.

To quantify the impact of time of day and weekdays on the download speed in the LTE network, we perform stationary measurements to isolate the changes that occur due to performance varying over time from the changes that occur due to performance differences at different locations.

In the following, we will discuss the results obtained using the LTE Cat 3 modem. We omit the results obtained with the LTE Cat 12 modem as we have generally observed similar behavior, albeit the obtained throughput was higher (up to 110 Mbit/s). The change of the signal quality parameters RSSI and SINR was small. More precisely, for all stationary measurements, the RSSI ranges from -51 dBm to -49 dBm and the SINR from 7 dB to 16 dB. In contrast, in the mobile case, the values of, e.g., the RSSI vary from -94 dBm to -29 dBm. Thus, the stationary scenario avoids the high variance of downlink throughput and signal quality due to location changes. All measurements were taken in a single cell in the B3 band with a 20 MHz bandwidth.

Every day for a duration of three weeks, we performed 30 alternating measurements for TCP and UDP at 12 a.m., 6 a.m., 12 p.m., and 6 p.m. In total, we collected more than 2000 measurements for each protocol. For our stationary TCP measurements, we have used TCP CUBIC as



(a) TCP and UDP throughput for 30 runs, mean and standard deviation

(b) TCP vs. UDP throughput in the interval from 3 to 4 s

Figure 3.9: Comparison of TCP and UDP throughput

our congestion control. Note that the behavior can differ when using other TCP variants. We discuss this further in Section 3.2.

We present a comparison of the transport layer protocols UDP and TCP in 3.1.3.1. Furthermore, in 3.1.3.2, we present results comparing TCP performance for different times of day and weekdays.

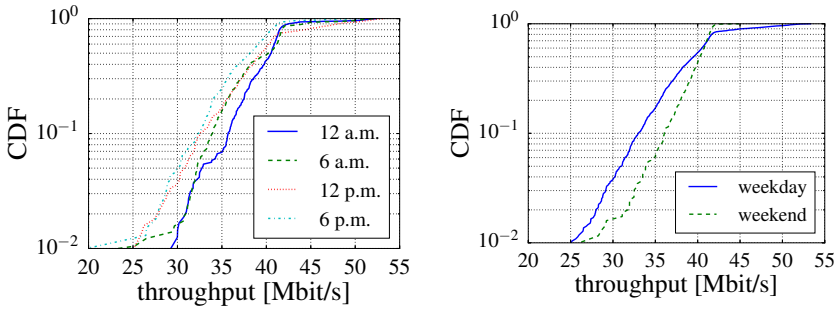
3.1.3.1 *Impact of the transport protocol*

As TCP and UDP are both widely utilized in mobile communication we want to investigate the potential differences in their downlink throughput performance in LTE.

In Fig. 3.9a, we show the average achieved download speed and the standard deviation of 30 alternating UDP and TCP measurements. It can be seen that after an initial phase where the congestion window of TCP is still growing, both protocols achieve the same rate. We observe this behavior for all measurements. Therefore, we conclude that in our measurements TCP congestion control does not affect the achievable throughput beyond the initial phase.

Next, we calculate the average throughput of the last second as described in Section 3.1.1 for all stationary measurements and present the results in Fig. 3.9b. As seen before, we obtain similar throughput for UDP and TCP, apart from about 2 percent of the TCP runs, where TCP retransmissions affect the download speed. In these cases, TCP retransmissions occur during the initial development of the congestion window, so that its size is heavily impacted for the rest of the measurement. However, when a retransmission occurs later, it rarely has any impact on the throughput. We also discuss this in more detail in Section 3.2.

Note that the Cumulative Distribution Functions (CDFs) of the TCP and UDP throughput have an exponential decline, meaning the download speeds are very stable and vary in most cases only within a small window.



(a) Time of day (b) Weekday vs. weekend

Figure 3.10: Comparison of different times and days.

From our experiments, we observe a stable development of the congestion window and that retransmissions at TCP layer are rare, which indicates an efficient layer 2 Hybrid Automatic Repeat reQuest (HARQ) implementation.

3.1.3.2 *Impact of the time of day & weekday*

In the following, we use TCP measurements to evaluate how the performance of the LTE downlink throughput changes over the course of a day and the week, to verify a potential impact of cell load changes over time.

In Fig. 3.10a, we show the variations of the measured download speed for different times. We observe that at night (12 a.m., 6 a.m.), the median throughput is slightly larger (41 Mbit/s) than at noon (39 Mbit/s) and at 6 p.m. (37.5 Mbit/s). The deviation increases marginally for 1 % of the runs. We have already identified retransmissions as the reason for this behavior in Section 3.1.3.1.

In Fig. 3.10b, we present the CDF of the throughput divided by weekday/weekend. Here, we find that the maximum throughput has been higher during weekdays than on weekends. However, the measurements performed on the weekend show a more stable behavior and, in the median, a higher download speed.

We conclude that given a fixed measurement location, the achievable downlink throughput varies slightly by the time of day and by the day of the week. In comparison to the mobile measurements, the impact is negligible.

3.1.4 *Conclusion*

In this Section, we have analyzed which parameters have the most impact on the achievable download speed in a major commercial LTE network. For this, we performed and evaluated long-term measurement campaigns in a stationary and a mobile environment, respectively. For the original paper, we have performed measurements with

an LTE Cat 3 modem. In this Section, we have supplemented them with LTE Cat 12 measurements and compared the results.

We have observed differences in urban and rural infrastructures: In cities, we have observed more cells in a higher frequency range with a shorter range, whereas there were more low-frequency cells with a long range on the highway. Due to the different availability of bandwidth in different frequency ranges, this has also resulted in a lower bandwidth in rural areas and higher bandwidth in urban areas. When comparing Cat 3 and Cat 12 measurements, we have observed that the bandwidth was sometimes higher when using Cat 12 due to the use of carrier aggregation.

As a result of different bandwidth availability, we have also observed a higher throughput in urban areas. Surprisingly, even though LTE Cat 12 enables higher throughput due to different MCS and higher bandwidth, we have observed, on average, similar throughput results for both Cat 3 and Cat 12 measurements along our measurement route.

Furthermore, we identified a strong impact of the signal quality, such as SINR and RSSI/RSRP, on the obtainable throughput. However, the relationship between the signal quality and the throughput is only stochastic. This means that while, on average, higher signal quality leads to higher throughput results, it is still possible to obtain low throughout for high signal quality and (relatively) high throughput for low signal quality, depending on the instantaneous cell load. We further discuss how this stochastic relationship can be used for stochastic throughput predictions in Chapter 4.

Additionally, we have shown that the impact of the weekday and time of day is small compared to signal quality and bandwidth.

Finally, the influence of different transport layer protocols, i.e., UDP and TCP, is negligible after TCP has ramped up its congestion window after slow start. We further discuss how the choice of different TCP congestion control protocols can mitigate the effect of TCP's startup phase in mobile networks in Section 3.2.

3.2 TCP CONGESTION CONTROL IN CELLULAR NETWORKS

One of the biggest challenges in networks is the fair and efficient distribution of resources. This problem is challenging because of the stochastic nature of network traffic: Any number of users can send any amount of data at any time. One way this challenge is being solved is the use of TCP Congestion Control, a mechanism that tries to adjust the sending rate of a sender so that it gets a fair share of the available bandwidth without overloading the network. Since TCP congestion control is also used in cellular networks, in this Section, we first want to give a brief overview of what congestion control is and what types of it exist. Afterward, we will discuss why some of the usual congestion control methods can experience problems in cellular networks, what can be done to mitigate these problems, and which congestion control algorithms are better suited for mobile networks than others. This Section is based on [33]. It includes additional measurements with TCP Vegas, an evaluation of an additional scenario where the channel's capacity is continuously increasing, and further explanations and evaluations.

3.2.1 *TCP Congestion Control Basics*

Upper-layer protocols like TCP usually have no knowledge of the underlying network structure. This means that TCP congestion control algorithms can only make limited assumptions when trying to recognize signs of congestion. Nevertheless, without making any assumptions, it is also impossible to recognize network congestion. Therefore most algorithms try to make some simple assumptions about the causes and consequences of congestion. These algorithms usually assume that there is a bottleneck link with a buffer somewhere in the network responsible for potential congestion. Different algorithms then use this assumption in different ways:

LOSS-BASED ALGORITHMS Here, the algorithms assume that when the network is congested, the fill level of the buffer of the bottleneck link starts to increase. As the size of this buffer is limited, it starts to drop packets when it becomes too full. Loss-based algorithms assume that any observed packet loss likely results from this and is a sign of network congestion. The popular TCP Reno [34] and CUBIC [35] congestion control algorithms are loss-based.

DELAY-BASED ALGORITHMS Similar to loss-based algorithms, delay-based algorithms try to identify congestion by inferring the state of the bottleneck link's buffer. However, instead of waiting for packet loss to occur, they try to prevent it by trying to identify when the buffer fill level starts to increase by measuring the delays. When the

delays increase above a certain threshold, it is considered a sign of congestion. TCP Vegas [36] is an example of a loss-based algorithm.

HYBRID Hybrid congestion control algorithms combine the loss- and delay-based approaches. Compound TCP (CTCP) [37] is an example of such an algorithm: it has both loss-based and delay-based windows that are handled independently. The function of these windows is based on TCP Reno and TCP Vegas, respectively.

TCP Bottleneck Bandwidth and Round-trip propagation time (BBR) [38] is often mentioned when talking about hybrid congestion control algorithms, however in its first version it does not react to packet loss but instead tries to be aware of the fill level of the bottleneck buffer by always draining it after filling it in a probing phase when the observed delays increase. TCP BBRv2, the successor of BBR that is currently still in development, also decreases its congestion window when it observes packet loss that exceeds a certain threshold.

Most such algorithms treat the network as a black box and assume that packet losses and delays are likely caused by network congestion (TCP BBR being a notable exception). In mobile networks, the bottleneck link is typically located between the user equipment and the cell tower. The per-user capacity of this link is often limited by the signal quality of the link between the user equipment and the cell tower. Packet losses and excessive delays are usually caused by the wireless channel, not network congestion.

In the following, we look at how well different congestion control algorithms perform in mobile cellular networks. For this, we perform stationary measurements to analyze their behavior when the per-user capacity remains constant due to only minor changes in signal quality. We also perform mobile measurements to analyze what happens when the channel's capacity changes due to a change in signal quality.

3.2.2 *Methodology & Measurement Setup*

Our measurement setup is the same as described in Section 3.1.1. All measurements have been performed with the LTE Cat 12 modem. The client uses Ubuntu 19.04 as its operating system. The server uses Ubuntu 19.04 for Reno, CUBIC, and BBR. For CTCP measurements, Windows Server 2016 is used. For all congestion control algorithms, we have used the default settings provided by the respective operating systems. On Ubuntu, we use TCP tracepoints to record the congestion window. On Windows Server, as the congestion window information is not easily accessible, we use Wireshark traces to analyze the number of bytes in flight. In our Ubuntu measurements, this number has been identical to the size of the congestion window.

In the stationary measurement campaign, we perform greedy throughput downlink measurements with different congestion control algo-

rithms to assess their performance in the absence of major changes in capacity. The corresponding results are described in Section 3.2.3.

In the mobile scenario, we send traffic from our test server to the user equipment while repeatedly driving on a city and highway route between Hanover and Brunswick with a speed of 100-130 km/h as described in Section 3.1.1. The results can be found in Section 3.2.4.

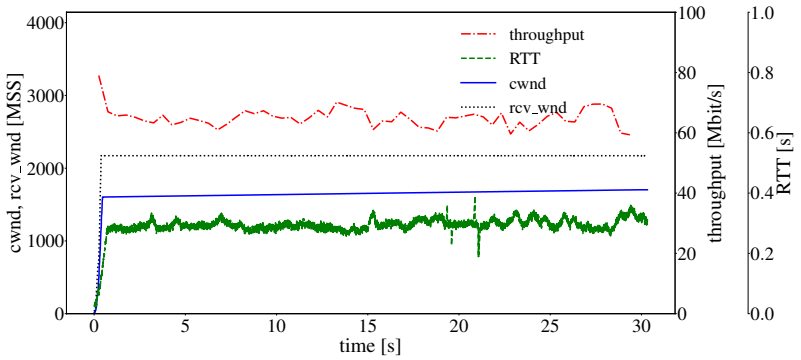
3.2.3 *Stationary Measurements*

The primary purpose of performing stationary measurements before mobile measurements is to isolate effects introduced by the congestion control algorithm from the effects resulting from signal quality varying with location and time/load-based effects. For this, we have performed multiple consecutive TCP throughput measurements while alternating between the congestion control algorithms. In the following, we will first present the behavior we have observed most frequently: In most connections, no packet loss could be observed. We then will briefly discuss the less frequent cases in which packet losses did occur.

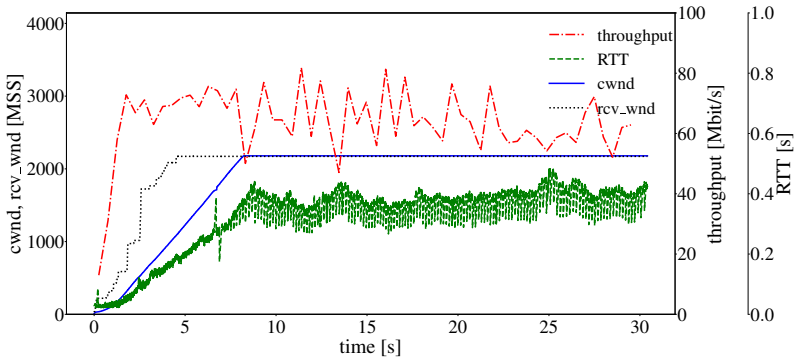
3.2.3.1 *Loss-based congestion control*

In this Section, we analyze the behavior of loss-based congestion control algorithms in a stationary scenario. For this, we choose Reno and CUBIC as the two most prevalent loss-based congestion control algorithms [39]. The main difference between these two algorithms is their congestion avoidance phase: Both algorithms increase their congestion window when Acknowledgement (ACK) packets arrive. However, Reno uses a linear function to increase its window, whereas CUBIC uses a cubic one.

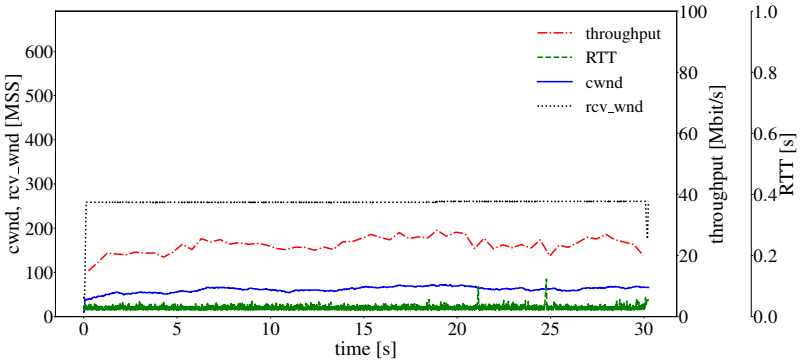
Both algorithms use the Additive Increase, Multiplicative Decrease (AIMD) scheme: When in the congestion avoidance phase, they carefully probe for new bandwidth by slowly increasing their congestion window. When a packet loss occurs, they assume this happened due to congestion. In order to allow other stations to increase their congestion windows, they reduce their congestion window by half (or to its initial state if Reno has identified the loss via a timeout) and start slowly increasing it again. This process finally converges to all stations having a similar congestion window. Currently, Reno (in its NewReno variation with some minor adjustments) and CUBIC are some of the most used congestion control algorithms. Additionally, CUBIC is often deployed in combination with the Hybrid Start (HyStart) mechanism, which tries to exit slow start before a packet loss occurs [40].



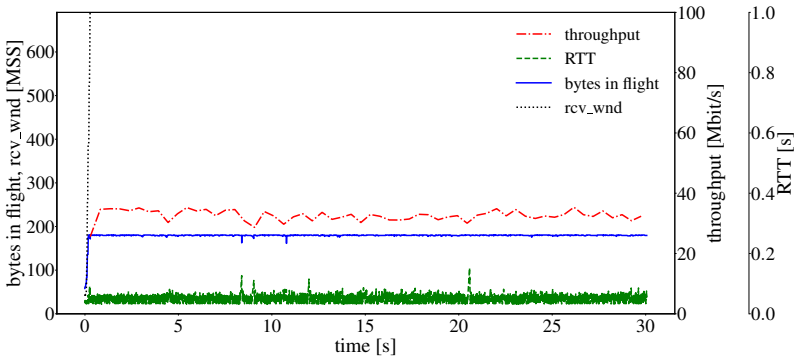
(a) Reno



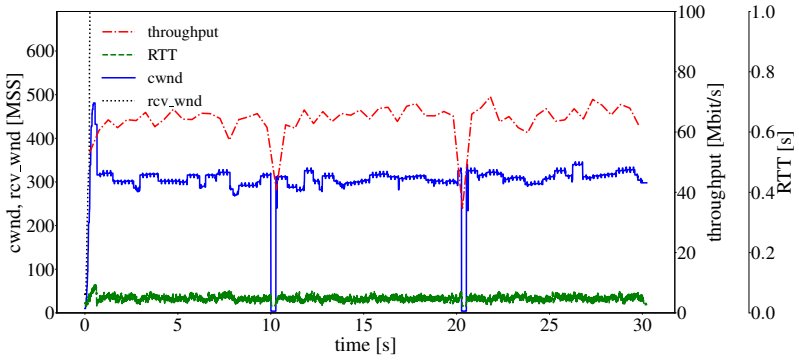
(b) CUBIC



(c) Vegas



(d) CTCP



(e) BBR

Figure 3.11: Stationary greedy throughput measurements with different TCP congestion control algorithms. The Maximum Segment Size (MSS) is 1448 Bytes. Note that the cwnd axis is scaled differently in different figures due to large differences in the behavior of the algorithms.

Figs. 3.11a and 3.11b display the results of two 30-second long greedy throughput measurements using Reno and CUBIC, respectively. The two measurements were performed right after each other, so the signal quality conditions and the available capacity are nearly identical. The only difference in the setup is the choice of the congestion control algorithm; no other parameters have been changed.

In both cases, it can be seen that the congestion window, after an initial phase of rapid growth, enters a phase where it either grows very slowly or even remains constant. In the case of Reno, the initial growth phase is the slow start phase, in which the congestion window is doubled each Round Trip Time (RTT). In the presented measurement, the threshold value of approximately 1600 MSS is reached very quickly, and the exponential growth is then stopped. After this, the window continues to grow linearly; however, the growth rate is very small compared to the already oversized window.

In the case of CUBIC, the initial growth phase is much longer than that in the case of Reno. This is because the default implementation of

CUBIC in Linux has the HyStart feature activated: With this additional algorithm, CUBIC can exit the slow start phase early if it detects congestion using the gaps between the ACK packets [41]. As already discussed in Section 2.1.2.1, the data transmitted over the LTE wireless channel is divided into Transport Blocks with a duration of 1 ms. These Transport Blocks are then scheduled individually, meaning that data packets that have arrived at the eNodeB in a single burst can be divided into multiple Transport Blocks and be scheduled with a time gap. On the other hand, the scheduler can combine multiple packets with a larger packet gap into one Transport Block so that the packet gap between them decreases. Because of this, the gaps between packets and ACKs can become bigger as well as smaller without it being a result of congestion. For this reason, HyStart almost always exits the slow start phase at the very beginning of the connection. A similar result has also been reported in a previous simulation study [42]. Because of this, the congestion avoidance phase is entered very early, and the growth of the congestion window is much slower than that of Reno. With HyStart turned off, the start-up behavior of CUBIC is identical to Reno.

After the slow start phase is finished, the congestion window grows using CUBIC's congestion avoidance function. After about 8 seconds, the cubic growth is stopped when the congestion window has reached the size of the receive window (rcv_wnd). The congestion window then stays at that value for the remainder of the connection.

The most notable detail of both connections is that neither of them experiences any losses in a 30-second interval. Due to large buffers in cellular networks and efficient HARQ retransmissions, the packet losses are typically concealed to TCP congestion control. It can therefore be impossible for loss-based TCP congestion control algorithms to adjust their window to the actual available capacity, as they expect the network to start dropping packets if the congestion window is too large.

The result of this behavior is an increase in delays. While the RTT of an ICMP ping packet without any load rarely exceeds 50 ms, the RTT of the TCP packets during a greedy throughput measurement can be much higher due to the self-induced bufferbloat caused by an oversized congestion window. In the case of Reno, the delays reached around 300-350 ms, and with CUBIC, they increased up to 450-500 ms due to an even larger congestion window.

There are multiple ways to alleviate this problem:

- Some network providers use TCP middleboxes that divide the single TCP connection between the server and the user equipment into two separate connections: One connection between the server and the middlebox in the LTE backbone network and one connection between the middlebox and the user equipment [6]. This way, the application server can still maintain a TCP con-

nection where the bottleneck link is not the wireless connection between the eNodeB and the user equipment so that the congestion control algorithm behaves properly. The middlebox can then manage the connection to the user equipment in a way that is aware of the limitations of TCP congestion control algorithms in LTE connections.

- Some smartphone manufacturers simply reduce the maximum receive window based on the type of communication/expected capacity so that the congestion window cannot become too large [43].

3.2.3.2 *Delay-based congestion control*

As we have seen in the previous Section, when no packet loss occurs in an LTE connection, the congestion control algorithms cannot be informed about network congestion, which can lead to performance issues. As delay-based congestion control algorithms adjust their congestion window before packet loss occurs, it can be reasonably assumed that they have an advantage when it comes to mobile networks.

In this Section, we take a look at TCP Vegas and CTCP, two well-known congestion control algorithms that use delay information when adapting the size of the congestion window, to see if they perform better than the loss-based algorithms.

TCP Vegas is purely a delay-based congestion control algorithm. It saves the lowest RTT observed as *baseRTT* and uses it to calculate the expected throughput based on the current size of the congestion window. The congestion window is reduced if the actual throughput is lower than expected.

An example of typical behavior with TCP Vegas in a stationary scenario can be seen in Fig. 3.11c. Here, it can be seen that after the initial, quick ramp-up during the slow start phase, the congestion window remains very low (note that the cwnd-axis is scaled differently from Reno and CUBIC). This also results in a much lower throughput than that achieved by TCP Reno or CUBIC. The most likely cause for this behavior is how Vegas estimates the expected throughput: This estimate is calculated using the lowest measured RTT value. Because of how the scheduling works in mobile networks, this estimate can be much lower than what would be representative of the connection, leading to a throughput expectation that is too high. Since this expectation can not be met, TCP Vegas keeps the congestion window low. Because TCP Vegas fails to fully utilize the link, the delays also remain low. We have also observed an alternative start-up behavior where TCP Vegas first grows a slightly larger congestion window and then gradually reduces it to a similar value as shown here over several seconds; an example of it can be seen in Fig. 3.13c.

CTCP, which has been the congestion control algorithm of choice for many versions of Windows and Windows Server operating systems, uses an approach that combines a congestion window calculated using Reno and an additional delay window [37]. The idea behind it is to not only be able to adjust the congestion window during loss events but also react when an increase in queuing delays is detected. The concept behind the detection of queuing delays is based on the same idea as in Vegas, i.e., the current RTT and an estimate of the RTT without load are used to calculate how many packets are queued in the network. If the queue is too large, the delay window is reduced to 0. If the queue is small, the additional delay window increases the congestion window.

Fig. 3.11d shows the result of one 30-second long greedy throughput measurement using CTCP. Note that instead of the congestion window, the number of bytes in flight is displayed, which is, however, a good approximation of the congestion window. When comparing this figure to Reno and CUBIC, it can be seen that the achieved throughput is much lower, which is due to the very small congestion window. This result is similar to what we observed with TCP Vegas. It is unclear what exactly limits the growth of the congestion window in this case since the loss-based window should grow even when the growth of the delay-based window is impeded as it was with TCP Vegas. We are, however, unable to investigate this further due to the closed-source nature of Windows. Since the connection clearly underutilizes the link, we can also see that the delays are much lower than those of Reno and CUBIC, as no bufferbloat is occurring.

This result also means that both delay-based algorithms have underestimated the link capacity and have caused the throughput to be much lower than would be possible.

3.2.3.3 *Model-based congestion control*

As already mentioned, TCP BBR is a model-based congestion control algorithm that tries to find an optimal working point according to [44]. In its first version, BBR periodically tries to increase its window and uses delays as an indicator of whether this has led to a higher buffer fill level. In its second version, it also uses packet loss and Explicit Congestion Notification (ECN) signals; However, it defines a threshold that should not be crossed instead of reacting to every lost packet. In this thesis, we analyze BBR's first version, as the second one is still in an early development stage.

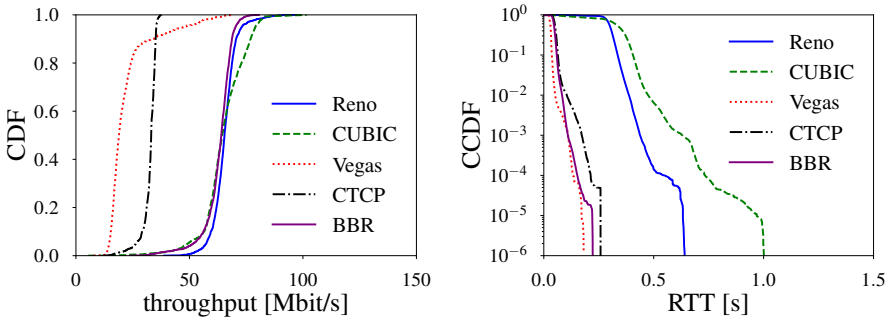
BBR works by trying to keep the fill level of the bottleneck buffer low while periodically probing the network for more available bandwidth. It increases its congestion window once every ten transmission rounds and then checks if this has increased the RTT. If it has not, the increased congestion window is kept; otherwise, it is decreased again. This way, BBR tries to avoid self-induced congestion while still probing for newly available bandwidth.

In Fig. 3.11e, which displays the result of a 30-second long greedy throughput measurement using BBR, it can be seen that it achieves a throughput similar to that of Reno and CUBIC while maintaining a much smaller congestion window. This also leads to much smaller delays since the network is not as overloaded. The only times when the throughput of BBR is reduced is during the ProbeRTT phases in which BBR reduces its congestion window to 4 segments to measure the RTT without load.

Overall, it can be seen that TCP BBR is the most successful in dealing with a mobile network where the bottleneck link is located between the user equipment and the cell tower, as it maintains a low delay and high throughput.

3.2.3.4 *Average behavior and packet losses*

For a statistical evaluation, we performed 120 alternating 30-second greedy throughput measurements with the five investigated TCP congestion control algorithms (30 measurements each). All the measurements have been performed consecutively so that the signal quality conditions remained nearly the same. Fig. 3.12a shows the distribution of the achieved throughputs, Fig. 3.12b the corresponding RTTs during the measurements. It can be seen that Reno, CUBIC, and BBR all achieve similar throughput. CTCP and Vegas, as discussed in the previous Section, fail to utilize the available capacity and thus achieve a much lower throughput. When looking at RTTs, we can see that BBR performs very similarly to CTCP and Vegas despite achieving much higher throughput. As both Reno and CUBIC achieve their high throughput by oversaturating the link, we can see that they both have much higher delays than CTCP, Vegas, or BBR.



(a) Throughput CDF

(b) RTT CCDF

Figure 3.12: Result of 150 alternating 30-second long greedy throughput measurements with five different TCP congestion control algorithms (30 measurements for each algorithm).

We note that while the cases we have presented in Fig. 3.11 are the ones we have observed most frequently, many connections still

	Reno	CUBIC	Vegas	CTCP	BBR
connections with at least one packet loss event	13.8%	30.3%	3.3%	19.3%	12.1%

Table 3.3: Percentage of TCP connections that experienced at least one packet loss in 150 alternating 30-second long greedy throughput measurements (30 measurements for each algorithm).

experience minor packet loss, as seen in Table 3.3. In about 12-30 % of the cases (depending on the congestion control algorithm), one or two packet loss events occurred during the 30-second measurement period. For Reno, such events helped to reduce the delays: The congestion window was halved for the remainder of the connection, which improved the delay performance due to a decrease in bufferbloat. For CUBIC, the packet loss led to a temporary reduction of the congestion window by 1/3, which improved the delays slightly; however, it grew back to the size of the receive window within about 5-10 seconds. It can also be seen that CUBIC experienced the highest packet loss rate, which most likely happened due to self-induced congestion, as CUBIC also had the largest congestion window and the highest delays. The throughput of Reno or CUBIC was not affected by packet losses in any of the observed cases, as the reduced window was still larger than needed to utilize the full capacity of the bottleneck link.

CTCP, Vegas, and BBR connections also experienced packet losses. However, neither of the algorithms has reacted to packet loss in an obvious way where a major change in the congestion window/bytes in flight could be observed that could not be attributed to a momentary change of capacity.

We note that while our results have been consistent with the results in other studies like [45, 46, 47], LTE connections can behave differently from what we have observed in this paper: The study in [48], for example, reports much more frequent packet loss in TCP connections using LTE than what we have observed in our measurements. While we can not say with certainty what the cause of this difference is, possible reasons are different service providers or operating systems (We use Ubuntu 19.04 on our receiver, whereas this study uses Windows XP). The authors of [49] also report a higher loss rate; however, their study deals with TCP performance at speeds of 300-350 km/h, where losses occur more frequently [50].

3.2.4 *Mobile Measurements*

In this Section, we discuss how the choice of congestion control algorithm affects LTE performance in mobile environments. For this, we have performed several hundreds of greedy throughput TCP measurements for several months with each of the discussed algorithms

while commuting on a highway between Brunswick and Hanover, as described in Section 3.1.1. Instead of presenting a statistical comparison between the different algorithms, which has been done in previous works (e.g. [45, 49]), we look at how they behave in two typical, challenging scenarios:

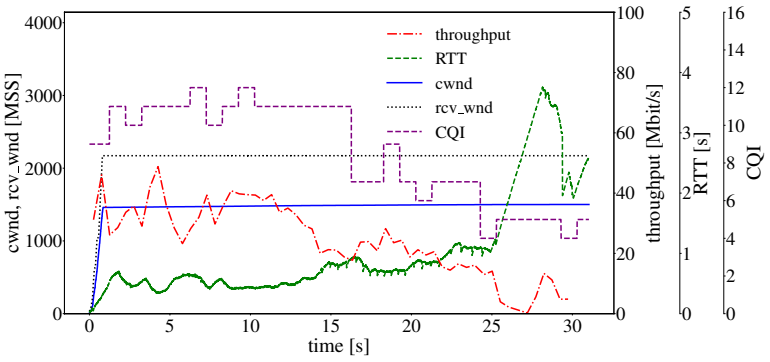
- A continuously increasing bottleneck link capacity due to improving signal quality conditions (e.g., while the user equipment is approaching an eNodeB)
- A decreasing bottleneck link capacity due to deteriorating signal quality conditions (e.g., while the user equipment is moving away from an eNodeB)

We use the CQI value reported by the modem as the indicator of channel capacity as it has a direct impact on the selected MCS, and therefore the maximum throughput, see Section 2.1.4. Note that CQI is not the only factor that influences the achieved throughput: other factors, like additional cell load caused by user equipment in other vehicles, can vary over time and are not considered in this study.

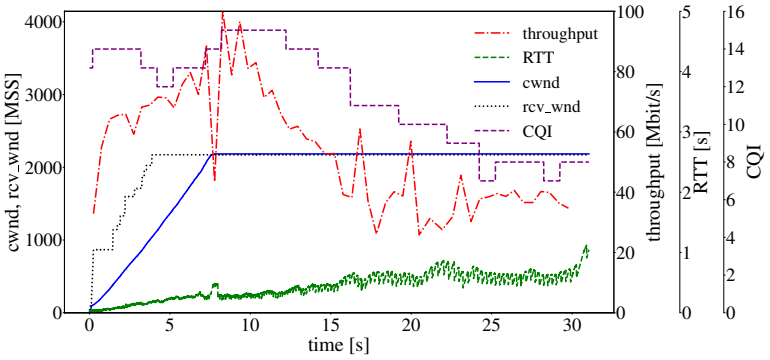
In Fig. 3.13, we present the results of five such measurements that have been performed using different congestion control algorithms. These measurements represent the typical behavior of these algorithms that we have observed throughout the measurement campaign; we have merely selected such cases in which the signal quality change is continuous and not erratic. Furthermore, we have selected measurements in which packet loss does not occur. As we have already discussed in the previous Section, this is the typical behavior we have observed in most of our measurements.

3.2.4.1 *Decreasing Capacity*

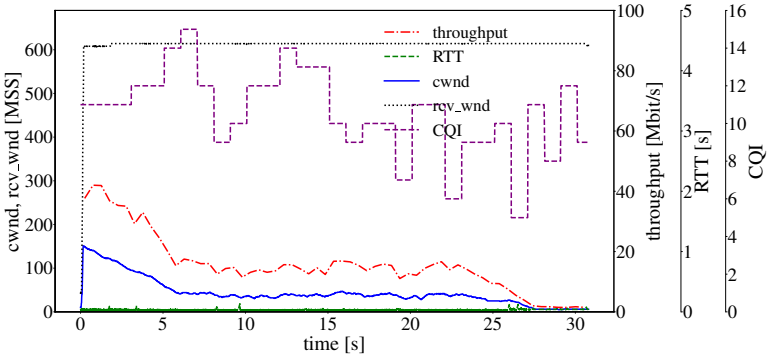
A continuously decreasing channel capacity can be a challenging scenario for congestion control algorithms: As the capacity decreases, old estimates of the channel capacity become invalid, and the congestion window has to be reduced. Since this reduction of the congestion window also has to happen continuously, particularly loss-based algorithms can be at a disadvantage since they probe for new bandwidth by increasing their congestion window.



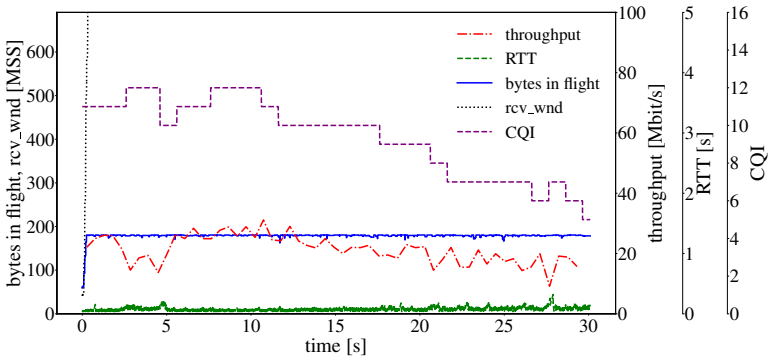
(a) Reno



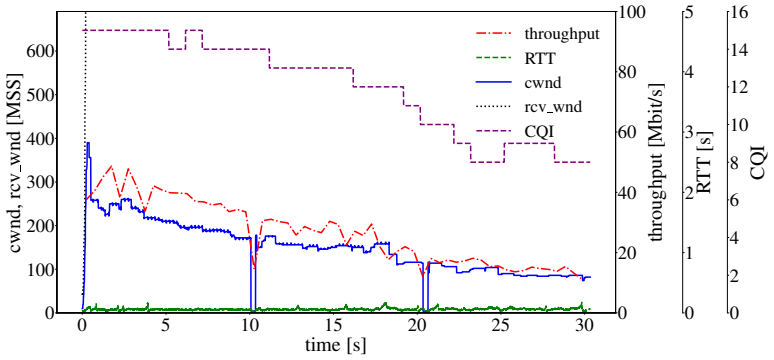
(b) CUBIC



(c) Vegas



(d) CTCP



(e) BBR

Figure 3.13: Mobile greedy throughput measurements with different TCP congestion control algorithms in which a continuous decrease of signal quality can be observed. The MSS is 1448 Bytes. Note that the congestion window (cwnd) axis is scaled differently in different figures due to large differences in the behavior of the algorithms. Also, note that the RTT axis is scaled differently than in Fig. 3.11 as delays can be much higher in mobile measurements.

LOSS-BASED CONGESTION-CONTROL In Fig. 3.13a, we look at how Reno reacts when such a decrease in capacity occurs. Before the decrease, Reno behaves similarly to the stationary case: the congestion window quickly reaches the threshold value and then continues to grow very slowly. The difference can then be observed at about the 15-second mark, where the CQI starts to decline continuously. Immediately we can see that the throughput drops, and the delay also begins to grow slightly. As the signal quality and throughput decline even further towards the 25-second mark, the RTT grows to 4 seconds as the buffer can no longer be processed in a timely manner. As no packets are lost, the delays only improve as the throughput slightly increases at the very end of the experiment. Note that no such delay variations were observed in the stationary case since the capacity has remained constant throughout the measurement.

The behavior of CUBIC in Fig. 3.13b is very similar: At the start of the connection, we see the same long transient phase as in the stationary measurement. After the congestion window reaches the size of the receive window, its growth stops, and it stays at that value until the end of the measurement, which is the same behavior we observed in the stationary case. The main difference can be observed around the 13-second mark: As the signal quality decreases, so does the throughput. Since no packet loss occurs, the congestion window remains unaffected by the decrease of capacity, which results in delays growing to 1 second toward the end of the connection. This differs from the stationary case, where the delay and capacity remained constant.

In both cases, we can see that when the channel capacity decreases, the congestion window is not reduced by a packet loss. This leads to increased delays since more time is needed to process the same amount of packets in network buffers. As this behavior has been what we have generally observed in all of our measurements, it can be said that both Reno and CUBIC are not suited for mobile scenarios in which the channel capacity is reduced, as is the case when the user equipment is moving away from an eNodeB.

DELAY-BASED CONGESTION CONTROL The behavior of TCP Vegas can be seen in Fig. 3.13c. At the beginning of the connection (0-5 second interval), we can see a behavior that we have also observed in many stationary TCP Vegas connections, i.e., the congestion window is reduced even though the channel capacity remains constant, resulting in throughput becoming lower. In the 5-25 second interval, the capacity of the channel experiences increases and decreases; however, this does not seem to affect the congestion window as it remains on the same level. The throughput also remains the same as it is limited by the congestion window. Finally, when the capacity of the channel experiences a significant decrease after the 25-second mark, we see a slight increase in RTTs. To account for that, TCP Vegas reduces the congestion window, resulting in reduced throughput. The RTTs become slightly lower again afterward. Overall it can be said that because TCP Vegas initially underestimates the channel capacity, it also does not react to its small changes. When the reduction of channel capacity becomes so large that it is below the initial estimate, the congestion window finally gets reduced.

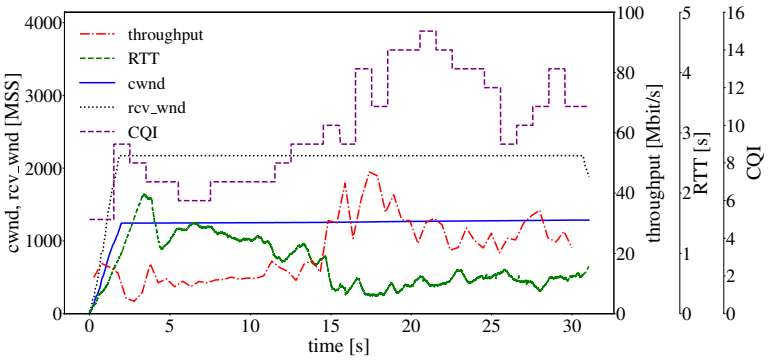
CTCP suffers from similar problems as the loss-based congestion control methods: As we have already seen in the stationary case, CTCP does not seem to make use of its delay-based window when using LTE. This results in a behavior similar to what we have observed with Reno and CUBIC: The behavior of the congestion window is identical to the stationary case despite the decreasing capacity. When the signal quality declines around the 20-second mark in Fig. 3.13d, the number

of bytes in flight does not change. Similar to what we have seen with Reno and CUBIC, no packet loss occurs to reduce the congestion window. Because of this, we can also see an increasing delay at the same time; however, since CTCP only grows its congestion window to a relatively small value that usually does not utilize the entire available capacity, this increase is very small. Note that such an increase in delays could not be seen in the stationary case since the capacity remained the same.

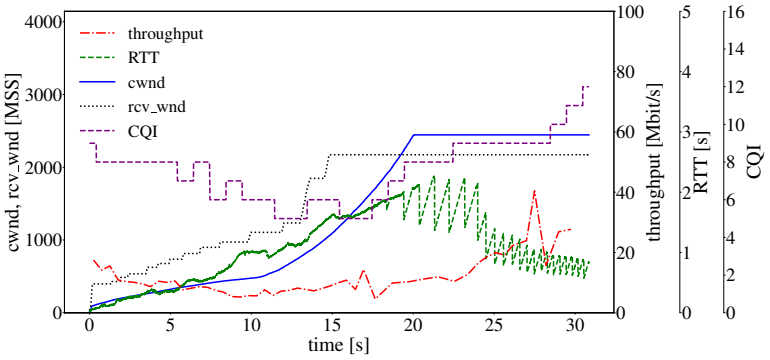
MODEL-BASED CONGESTION CONTROL BBR, unlike the other algorithms, was able to adjust its congestion window to a change in signal quality. This is because the changing capacity also changes the RTT delays, as we have seen in the other examples. Since BBR is designed to react to a change in delays, it is able to reduce its window when the capacity decreases. Because of this, BBR is the only one of the analyzed congestion control algorithms that exhibits a different behavior in the mobile scenario than it did in the stationary measurements: In Fig. 3.13e, we see that the slope of the congestion window follows the decreasing slope of the signal quality, instead of remaining nearly constant like in the stationary case. This results in the delays remaining low when the signal quality decreases instead of increasing like in the other cases. Note that BBR exhibits similar, nearly constant RTT delays in both the stationary and the mobile case.

3.2.4.2 *Increasing Capacity*

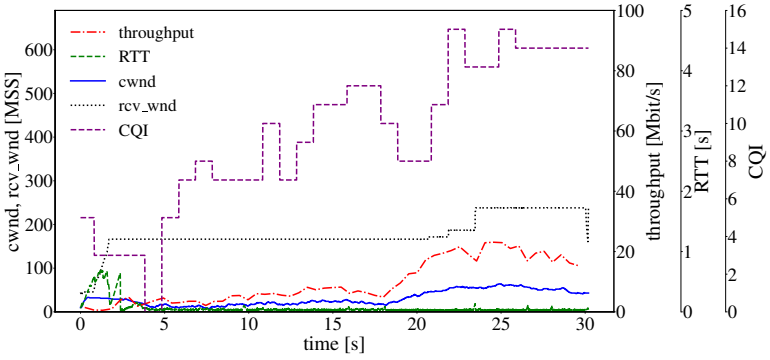
A continuously increasing capacity can also be a challenging scenario for congestion control algorithms. Even though congestion control usually continuously probes for new bandwidth, the rate at which the capacity increases can differ from the rate at which the congestion window grows. In this Section, we once again analyze selected traces from our mobile measurement campaign where the capacity of the channel, as indicated by the CQI, has been continuously decreasing.



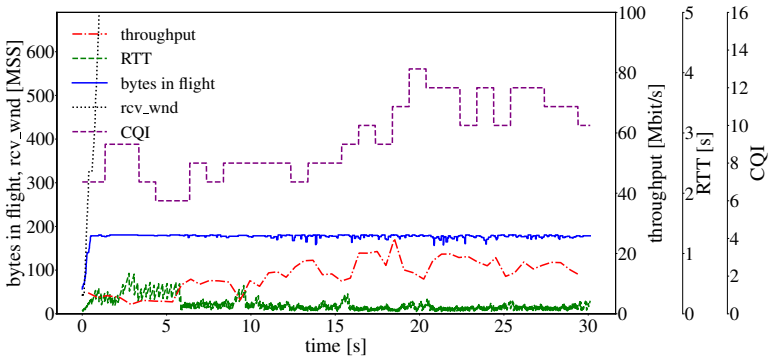
(a) Reno



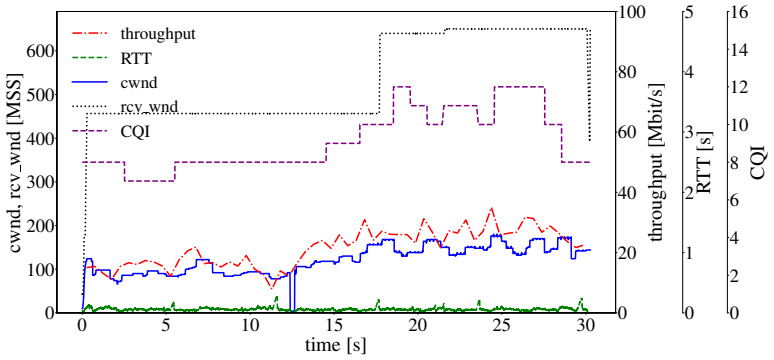
(b) CUBIC



(c) Vegas



(d) CTCP



(e) BBR

Figure 3.14: Mobile greedy throughput measurements with different TCP congestion control algorithms in which a continuous increase of signal quality can be observed. The MSS is 1448 Bytes. Note that the cwnd axis is scaled differently in different figures due to large differences in the behavior of the algorithms. Also, note that the RTT axis is scaled differently than in Fig. 3.11 as delays can be much higher in mobile measurements.

LOSS-BASED CONGESTION-CONTROL In Fig. 3.14a, we look at what happens when the capacity of the link increases while using TCP Reno. Before the capacity starts to increase around the 10-second mark, we once again see the same behavior as in the stationary case, where the congestion window first rises during the slow start phase and then only grows very slowly. We can also see that no packet losses occur to adjust the congestion window so that the RTT grows to 1-2 seconds. When the CQI starts to increase, we see that the throughput also starts growing. As increased throughput allows the bottleneck buffers to be processed faster, we also see a declining RTT. When the capacity decreases again around the 25-second mark, we again see that the throughput declines and the RTT slightly increases. The congestion window has the same slow growth throughout the connection regardless of the channel capacity.

The behavior of CUBIC is shown in Fig. 3.14b. Here we see that even though the congestion window is already large enough to support a throughput of about 20 Mbit/s that we see at the beginning of the connection, the congestion window continuously grows throughout the connection. After an initial decrease of capacity in the 0-5 second interval, we see that the CQI and the throughput remain more or less stable in the 5-15 second interval. Because of the growing congestion window, we can also see the delays grow. When the capacity increases after the 15-second mark, we see that the throughput also slowly starts to increase. Because the congestion window keeps growing, the RTT still increases. When the congestion window stops growing around the 20-second mark, the bottleneck buffer begins to be processed, and the RTT becomes smaller. Once again, the connection never experiences a packet loss to adjust the oversized congestion window.

In both cases, we see that an increasing channel capacity results in lower delays. However, it does not affect the growth of the congestion window. The congestion window usually reaches a large enough size to support the maximum throughput at the very beginning of the connection and keeps growing. While this case is not as problematic as the previous case, where the capacity was decreasing, it still shows that the loss-based congestion control algorithms cannot adjust their congestion windows to the capacity of the link.

DELAY-BASED CONGESTION CONTROL The behavior of TCP Vegas can be seen in Fig. 3.14c. At the beginning of the connection, we see that the capacity of the channel is very low. Nevertheless, Vegas starts the connection with a slightly oversized congestion window that is then reduced over time. This is the same behavior we have observed in both stationary and other mobile cases, meaning that this happens regardless of channel conditions. When the channel capacity increases in the 5-18 second interval, we see some small growth of the congestion window, which also results in increased throughput. When the channel capacity continues growing even further in the 18-30 second interval, we see even more significant growth of the congestion window and the throughput. This shows that TCP Vegas can adjust its congestion window when the channel capacity is growing. However, as this is a measurement in a mobile environment where the total channel capacity is unknown, we cannot say for sure whether TCP Vegas actually utilized the total available capacity.

The behavior of CTCP is shown in Fig. 3.14d. Once again, we see that CTCP keeps the number of Bytes in flight constant throughout the connection. When the channel capacity increases at the 5-second mark, we see that the throughput increases and the RTTs decrease, which is the same behavior we have observed with Reno. This means that in the first 5 seconds of the connection, the congestion window of CTCP was high enough to utilize the entire channel capacity. However,

since it did not grow afterward, it cannot be said if an even higher throughput could have been reached if it grew further.

MODEL-BASED CONGESTION CONTROL The model-based TCP BBR is able to adjust its congestion window in this scenario as well. In Fig. 3.14e, we see that when the channel capacity starts to increase around the 14-second mark, the congestion window and the throughput also start to grow. The RTTs remain at the same level both before and after the increase of the capacity/congestion window. Note that just like in the case of TCP Vegas, we cannot say for sure that TCP BBR is actually utilizing the entire available capacity since we do not know it for the mobile scenario.

3.2.5 *Conclusion*

Overall, it can be said that due to the fact that the bottleneck link in mobile networks is usually located between the user equipment and the eNodeB, congestion control algorithms often have the task of finding the capacity of that link instead of trying to prevent network congestion. Because of large buffers in the mobile network, the connection rarely experiences packet losses, meaning that loss-based algorithms are often unable to adjust their congestion window. Instead, these buffers get filled by the congestion control, which leads to very high delays. The lack of packet losses can also be problematic since the channel capacity can fluctuate rapidly and even decrease continuously in cases like when the user equipment is moving away from an eNodeB.

Congestion control algorithms with a delay-based component like TCP Vegas and CTCP do not cause bufferbloat like the loss-based ones. However, they are also very cautious when probing for newly available bandwidth. At least in our experiments, these algorithms were only rarely able to adjust their congestion window to reach the capacity of the wireless link. However, it should be possible to improve the performance of these algorithms for mobile networks by simply adjusting the way they calculate the expected throughput: If the algorithms consider that the lowest measured delay is not necessarily the lowest delay that occurs without load but instead suffers from high variability, the estimate of the expected throughput should improve and be more representative of the actually achievable throughput.

The model-based algorithm TCP BBR has been the only algorithm that has been consistently able to utilize the total capacity of the channel and react to an increase or decrease in capacity by adjusting its window and therefore maintaining low delays. Because of this, we recommend TCP BBR for use in mobile networks.

3.3 CARRIER AGGREGATION

Due to increasing demand for higher data rates, carrier aggregation has been introduced as an improvement for LTE. Carrier aggregation seeks to increase the available capacity by combining multiple component carriers of 1.4 to 20 MHz for a total of up to 100 MHz in Release 12 or up to 640 MHz in Release 13 [4]. To achieve this, an eNodeB capable of carrier aggregation can use its scheduler to assign transport block (TB) transmissions to multiple carriers instead of just one, thus increasing the transmission capacity. The UE has to receive the transport blocks from both component carriers and then reassemble them in the correct order before higher layers can process the data. Further details describing carrier aggregation implementation can be found in [4, 30]. Carrier aggregation will also be used in 5G; however, it plays a less important role as the frequency spectrum of 5G is less fragmented.

Since the use of multiple component carriers for data transmission creates additional complexity, we want to analyze whether the increased throughput comes with a trade-off in other areas. To answer this question, we perform multiple measurement campaigns in a commercial LTE network to analyze the possible impacts of carrier aggregation on the application-level performance.

Existing work analyzing the performance of carrier aggregation often focuses on comparing the average throughput with and without carrier aggregation and the relationship between throughput and signal quality. One such study presents measurement results of multiple test drives performed in South Korea [51]. Separate measurements are performed with and without carrier aggregation in 850 MHz and 2600 MHz frequency bands. The study shows that combining component carriers in the two bands using carrier aggregation allows to achieve a throughput that is, on average, about 97% of the sum of the average throughputs of two component carriers by themselves.

The authors of [52] use an LTE control channel decoder to analyze how users at three locations in Spain benefit from carrier aggregation. They find that while it is generally possible for users to achieve higher data rates using carrier aggregation, the user throughput requirements can often be met without it. In these cases, carrier aggregation is used as a load-balancing technique between the component carriers.

The authors of [53] have used a simulation to analyze the throughput achievable when using LTE carrier aggregation with 4x4 MIMO and compared the simulation results with real-world measurements. They have found that the correlation between the propagation paths in real-world measurements leads to a much lower throughput than the simulation predicts for statistically independent channels.

Many other works like [50, 20, 45, 49] analyze achievable throughput performance in live LTE networks; however, they do not comment on the availability of carrier aggregation and its effect.

In this Section, which is based on [54], we analyze how the use of carrier aggregation affects the application level performance of LTE. We describe the improvement that is achievable in a mobile scenario on a highway, where the user equipment faces challenges like constantly changing signal quality (and therefore capacity), handovers, and secondary component carriers being turned on/off during an active connection. Unlike other studies, we not only compare the average throughput performance but also analyze the effect of carrier aggregation on the round-trip-time delays and present the behavior that can be observed during the activation and deactivation of a secondary component carrier.

We have also performed stationary measurements in controlled signal quality conditions for a more detailed, statistical evaluation of the achievable improvements. We performed greedy UDP throughput measurements with and without the use of carrier aggregation. In contrast to other studies, we have not only analyzed the average throughput improvement but also how the throughput that can be achieved at a particular time in the individual bands is related to the aggregated throughput when using multiple component carriers. This evaluation is also important for performance prediction that we discuss in Chapter 4 as we need to know whether connections with a specific bandwidth need to be treated differently depending on whether this bandwidth has been achieved with or without the use of carrier aggregation.

Additionally, we have performed measurements with CBR traffic sent at different rates with and without carrier aggregation. We use these measurements to present the effect of carrier aggregation on the achievable delays, which can be negatively affected by a component carrier with a higher loss rate but also improved due to load balancing.

3.3.1 *Methodology & Measurement Setup*

Our hardware and software setup is the same as described in Section 3.1.1. When performing mobile measurements, we use TCP BBR as our congestion control algorithm as we had found in Section 3.2 that it adjusts its congestion window when the capacity of the link changes, unlike other popular algorithms that often kept the same congestion window throughout the connection. The measurements are performed in bands B3 and B20.

For the stationary throughput evaluation, we perform alternating greedy throughput downlink measurements with UDP in the B3 and B7 bands individually, as well as these two bands combined with carrier aggregation. We use iperf3 as our traffic generator. For the stationary

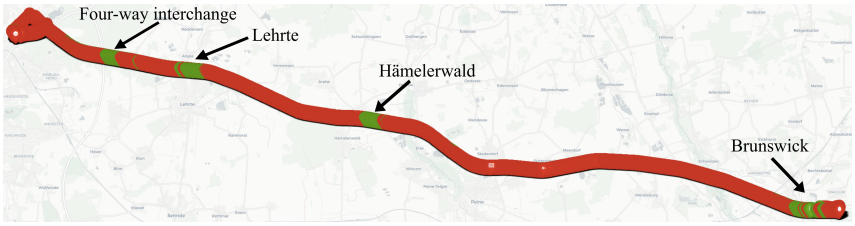


Figure 3.15: Map indicating availability of carrier aggregation (green = available, red = not available). In our measurements, carrier aggregation was only available close to populated areas or an interchange where more traffic is expected. Map data © OpenStreetMap contributors, CC BY-SA [26].

One-Way Delay (OWD) measurements, we use a python-based CBR UDP traffic generator.

We use different bands in the stationary and mobile measurements since the B7 band is more commonly used in urban areas where users tend to be stationary more frequently, and the B20 band is more common in rural and highway areas where users are more mobile [20].

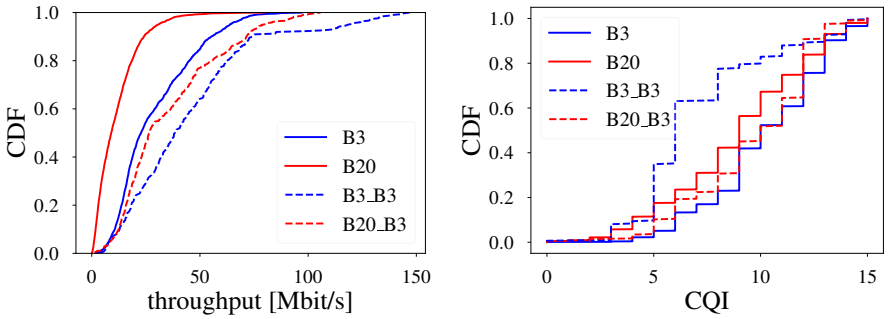
3.3.2 *Mobile Measurements*

In this Section, we discuss how the use of carrier aggregation affects LTE performance in a mobile environment. For this, we have performed multiple greedy throughput TCP measurements in Hanover and Brunswick and on the highway between the two cities. We have selected this highway since it is often used for daily commutes, and we wanted to analyze how carrier aggregation affects the performance of LTE communication in a frequently occurring scenario.

Since we cannot control the signal quality conditions in a mobile environment, we only evaluate the average throughput and delay performance in different bands with and without the use of carrier aggregation. Additionally, we present sample traces that show the behavior of the TCP connection when an additional component carrier is activated or deactivated while driving on a highway.

We compare mobile highway measurements in the bands B3 and B20 since the B7 band is rarely available outside the inner city due to its lower range.

We also note that carrier aggregation was very rarely spatially available in our measurements: As can be seen in Fig. 3.15, carrier aggregation was usually only activated close to populated areas or areas where more traffic is expected. We have also observed a similar result in a previous study [20]: LTE coverage was more sparse in the highway scenario, most likely due to cost-saving measures by providers in less densely populated areas. During our measurements on the highway, we observed that carrier aggregation was activated with component carriers in the bands B20 (10 MHz) and B3 (20 MHz),



(a) (b)

Figure 3.16: Throughput and CQI values of highway measurements.

which we denote as B20_B3 (also known as CA_3A-20A), as well as within the B3 band, where a 10 MHz carrier and a 20 MHz carrier were combined, which we denote as B3_B3 (also known as CA_3C). We have not observed carrier aggregation with three or more carriers in any of our mobile or stationary measurements.

3.3.2.1 *Mobile Throughput Measurements*

In Fig. 3.16a, we present a CDF showing the distribution of throughput values achieved with and without the use of carrier aggregation. In this CDF, it can be seen that, in the median, a higher throughput has been achieved with carrier aggregation (B3_B3: 38.5 Mbit/s, B20_B3: 26.8 Mbit/s) than without it (B3: 22.8 Mbit/s, B20: 9 Mbit/s), which can be explained by the additional available bandwidth.

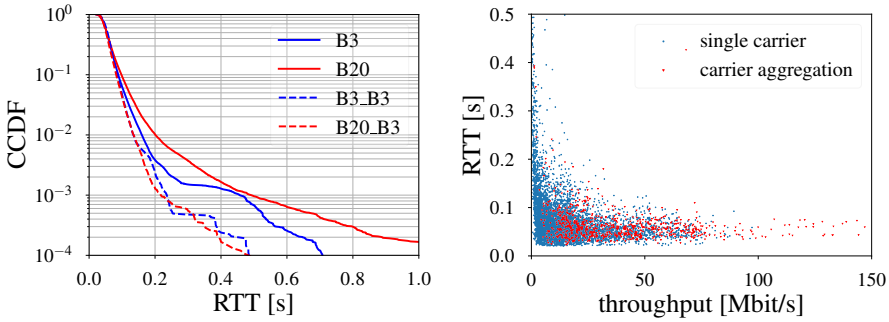
In Fig. 3.16b, we can see that for the B3, B20, and B20_B3 measurements, the signal quality values were similar to each other; however, they were lower for the B3_B3 measurements, meaning the higher median throughput has been achieved despite the lower signal quality.

Overall, our measurements show that connections that used carrier aggregation achieved a higher throughput, even when the signal quality was lower. In order to quantify the exact throughput improvements achievable when using carrier aggregation in similar signal quality conditions, we additionally perform stationary measurements, the results of which we present in Section 3.3.3.1.

3.3.2.2 *Mobile Delay Measurements*

For the delay evaluation in the mobile scenario, we take RTT measurements, as precise synchronization required for OWD measurements is hard to achieve when not in the same local area network.

In Fig. 3.17a, a Complementary Cumulative Distribution Function (CCDF) of all mobile delay measurements with and without carrier aggregation can be seen. While the median is almost the same for



(a) CCDF of mobile delay measurement results

(b) Throughput vs. RTT

Figure 3.17: Mobile delay measurement results.

measurements with and without carrier aggregation, the tails of the delay measurements with carrier aggregation decay faster.

The reason for this can be seen in Fig. 3.17b, where we present how the RTT depends on the achieved throughput. It can be seen that the delays increase when the throughput is low since the network buffers cannot be processed fast enough. It can also be seen that the delays are lower when the throughput is higher.

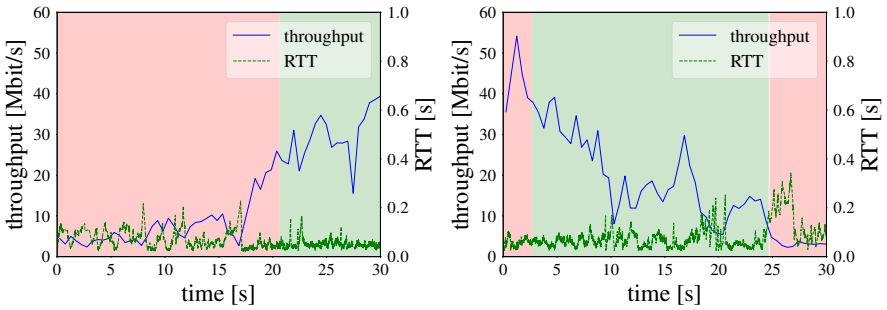
Since higher throughput occurs more frequently when carrier aggregation is activated, as we have seen in Fig. 3.16a, it can be said that carrier aggregation can lead to lower delays. We further investigate the relationship between throughput, delays, and utilization in Section 3.3.3.2 by performing measurements in a controlled, stationary environment.

3.3.2.3 *Handover Behavior*

In Fig. 3.18, we present two typical examples of the behavior we observed when carrier aggregation had been activated or deactivated during an ongoing connection. The state of carrier aggregation is indicated by the background (green=on, red=off). Note that the firmware of the modem has sometimes indicated the changes in the carrier aggregation activation state with a slight delay so that the actual switch might have happened slightly (1-2 seconds) before the time indicated in the figure.

In Fig. 3.18a, we can see that when carrier aggregation is activated, the throughput quickly starts growing because of the increased capacity, while the delays decrease. We did not observe delay spikes or similar behavior during the switch, i.e., the improvement of throughput and delays was the only observed effect.

In Fig. 3.18b, one of the carriers is switched off towards the end of the connection. We can see that the throughput has already been declining since the beginning of the connection. This is because the



(a) carrier aggregation is activated (B20 → B20_B3)

(b) carrier aggregation is deactivated (B3_B3 → B3)

Figure 3.18: Behavior when carrier aggregation is activated (green background) or deactivated (red background).

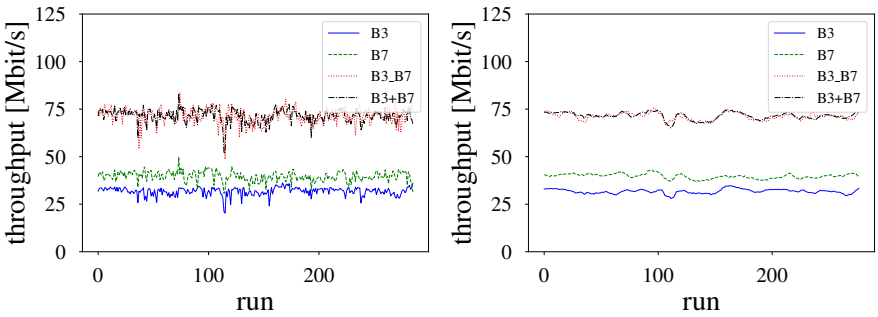
user equipment was moving away from the eNodeB. As we have observed during stationary measurements (see Section 3.3.3.2), when the throughput requirement is low, the eNodeB does not use a secondary carrier, which is also likely the reason for it being turned off when the throughput is decreased. After the second component carrier has been turned off, the throughput decreases further, and the delays also begin to grow.

We note that this behavior is different from the results described in [55], where the authors also analyze TCP RTTs with and without carrier aggregation in a live LTE network. Their measurements show that delay spikes occur more frequently while using carrier aggregation, which the authors explain with more frequent Radio Link Control (RLC) layer retransmissions. The difference may come from different providers and hardware used in the measurements.

3.3.3 *Stationary Measurements*

The primary purpose of performing stationary measurements with carrier aggregation is to isolate its effects from the effects that result from signal quality varying with location, as well as time and load-based effects. For this, we have performed multiple consecutive greedy throughput UDP measurements while alternating between component carriers in the B3 and B7 bands (20 MHz each), as well as their aggregation, which we denote as B3_B7 (also known as CA_3A-7A).

In Section 3.3.3.1, we analyze how the use of carrier aggregation affects the throughput by repeatedly performing greedy throughput measurements and comparing the performance in different bands. In Section 3.3.3.2, we analyze the delay performance by repeatedly sending UDP CBR traffic at different rates in alternating bands. The purpose of using different CBR rates is to compare delays under different loads.



(a) Throughput over time (b) Moving average over 10 measurements

Figure 3.19: Stationary throughput over time measured at the LTE receiver. The sum of the throughputs B3+B7 is strongly correlated with the CA performance.

3.3.3.1 *Stationary Throughput Measurements*

In Fig. 3.19, we present the result of 300 greedy throughput measurement experiments using UDP. Each measurement experiment consists of a measurement in the B3 band, a measurement in the B7 band, and a measurement using carrier aggregation of component carriers in the two bands. During each measurement, 5MB of data is transmitted from the server to the client with a sending rate of 100 Mbit/s. Each packet utilizes the entire MTU of 1500 Bytes. The measurements are performed within a few seconds of one another. Additionally, as a way to compare the performance of the individual component carriers and their aggregation, we calculate the sum of the two throughput values B3+B7 for each pair of consecutive measurements in these two bands. We use this value as a benchmark for the data rate that should be achievable using both bands simultaneously. Fig. 3.19a shows individual throughput values for each performed measurement.

As a way to smooth out the high variability of LTE throughput over time, we have also looked at the moving average of the measurements in each band which can be seen in Fig. 3.19b. In the figure, it can be seen that the sum of the throughput values in the two bands closely follows the throughput achieved during the use of carrier aggregation. This is also reflected in the high correlation (0.75) between the throughput achieved using carrier aggregation B3_B7 and the sum of the individual throughput values in bands B3+B7. The throughput measurement results in the bands B3 and B7 themselves are only weakly correlated (0.29), meaning that a performance change in one of them does not necessarily lead to a change in the other.

A comparison emphasizing the difference in throughput between carrier aggregation and the individual carriers can be seen in Fig. 3.20. This figure shows the CDF of all throughput measurement results using carrier aggregation compared to B3+B7. We did not differentiate

between the cases where the Primary Cell (PCell) was in B3 from those in B7, as the performance was nearly identical.

Overall, it can be said that in our measurements, carrier aggregation has managed to utilize the total capacity offered by the individual component carriers in the bands B3 and B7. This result is also important for performance prediction that we discuss in Chapter 4: it shows that connections that have a certain bandwidth due to the use of carrier aggregation, as well as those that do not use carrier aggregation, can be treated the same way when predicting the achievable throughput.

3.3.3.2 *Stationary Delay Measurements*

In addition to the achievable throughput performance, we have analyzed the delay performance with and without carrier aggregation under similar signal quality conditions. For this, we have transmitted CBR traffic at different rates using UDP and measured the OWDs from the server to the user equipment. In order to obtain precise OWD values, we have used the Precision Time Protocol to synchronize the sender and the receiver. For small sending rates of 1-2 Mbit/s, no activation of carrier aggregation could be observed; for other sending rates smaller than 5 Mbit/s, the activation was very infrequent. Therefore we focus our analysis on sending rates of 5 Mbit/s and higher. The available capacity in the B3 band was, on average, around 30 Mbit/s; in the B7 band it was around 40 Mbit/s. The results of these measurements can be seen in Fig. 3.21.

For low sending rates of 5-20 Mbit/s, the delays measured with and without carrier aggregation are very similar: The median is the same in each case, and the delay is less than 40 ms at the end of the tail (at 10^{-4}). The reason for this is that when the channel utilization is low, the delays mostly depend on the protocol delays, which are largely independent of the specific utilization [56].

The main difference can be observed in the packet loss rate. The packet loss rate can be seen in the CCDF by observing where the delays start to increase above the initial 10-12 ms. In Fig. 3.21a for a sending rate of 5 Mbit/s, it can be seen that the B3 band has a lower loss rate

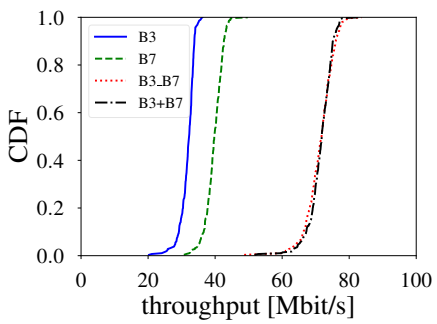
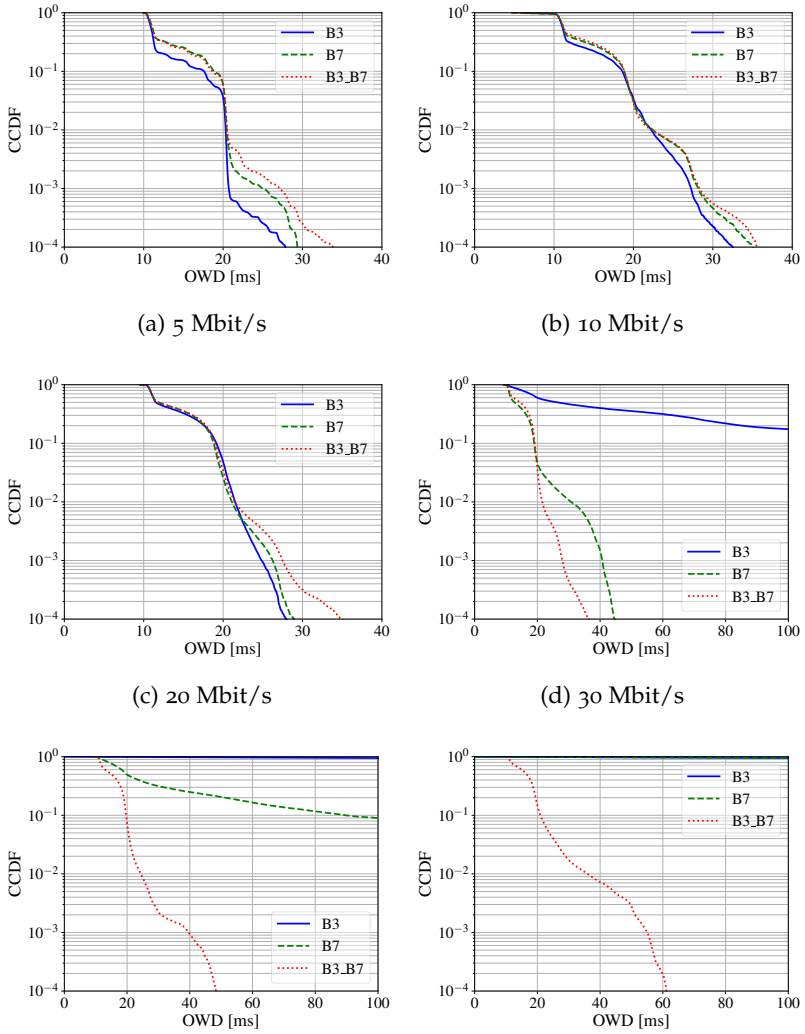


Figure 3.20: CDF of stationary throughput measurement results.



(a) 5 Mbit/s

(b) 10 Mbit/s

(c) 20 Mbit/s

(d) 30 Mbit/s

(e) 40 Mbit/s

(f) 50 Mbit/s

Figure 3.21: OWD measurements for UDP CBR traffic with different rates. Note that for 40 Mbit/s and 50 Mbit/s, some of the OWDs are so high that they are only visible at the top of the figure (at 10^0).

than the B7 band due to different channel conditions. The increased loss rate in the second band also leads to an increased combined loss rate when using carrier aggregation. As the loss rate in the B3 band increases with higher load this effect becomes less pronounced.

With increasing sending rates, the difference between the delay tails in the different bands becomes larger. In Fig. 3.21d, as the server sending rate is close to the capacity of B3, its tail decay becomes a lot slower, whereas the tail decay rate with the use of carrier aggregation is not affected. This is because the scheduler of the eNodeB balances the load between B3 and B7 when using carrier aggregation, leading to a decreased load in each band. The same effect can also be seen in Fig. 3.21e for the sending rate of 40 Mbit/s and band B7.

With higher sending rates of 50 Mbit/s and above, the delays when using carrier aggregation also increase as the sending rates approach the combined available capacity of B3 and B7. The delays increase with higher bitrates since the utilization is also higher [56].

Overall, it can be said that the use of carrier aggregation by itself, on average, does not significantly increase delays. However, it can impact the tail distribution as the component carrier with the higher loss rate will also slow down the other carrier since the transport blocks from both carriers must be processed in the correct order. For higher loads, carrier aggregation can improve the delay performance by offering additional capacity.

3.3.4 *Conclusion*

We have performed delay and throughput measurements with and without the use of carrier aggregation in a major commercial LTE network using the bands B3, B7, and B20. We found that the achievable throughput using carrier aggregation is nearly identical to the sum of the throughput values in the individual carriers. In contrast, the two carriers' throughput is only weakly correlated. This result is important for our performance prediction in Chapter 4, as this means that connections that use carrier aggregation do not need to be treated differently from those without it, and only the total bandwidth matters.

Our mobile highway measurements with carrier aggregation have shown a delay and throughput performance that was as good or better than without it. However, the spatial availability of carrier aggregation in highway areas was very low. An activation of carrier aggregation during an active TCP connection has always improved delays and throughput, whereas a deactivation had the opposite effect.

For OWDs, we found that the use of carrier aggregation does not introduce additional delays when the sending rate is low. However, if one of the carriers experiences higher delays due to a higher loss rate, carrier aggregation will also have higher delays and loss rate. When

the sending rate increases, carrier aggregation exhibits lower delays since the load is balanced between the individual carriers.

Overall, it can be said that from the application level point of view, the use of carrier aggregation can lead to an improvement of throughput and delays. The use of carrier aggregation can, however, negatively affect delays if one band experiences a higher packet loss rate than the other and the utilization is low.

4

PERFORMANCE PREDICTION

As we have discussed in the previous Chapter, there are many aspects that influence the data rate of an LTE connection and can be used for prediction:

- The hardware of the user equipment and the eNodeB has a major influence on achievable data rates. As discussed in Section 2.1, LTE has undergone major improvements throughout the years; however, not all user devices are up to date. Features like carrier aggregation and higher-order modulation can significantly influence achievable data rates.
- The user's signal quality significantly influences the achievable data rates. However, it cannot be directly used to predict the data rates as there are other influences, e.g., the number of other users in the same cell.
- The band in which the cell is located and the associated bandwidth can also be used to characterize the achievable data rates.
- The location of the user equipment can also have a high prediction value: rural areas usually have less coverage and lower throughput than urban areas; however, they also have a more stable throughput that does not vary as much.

In this Chapter, we devise a model that can be used to predict the achievable throughput at a specific location without performing active measurements. Note that due to the stochastic nature of the relationship between passive indicators like signal quality and throughput, the predicted value can also only be stochastic.

4.1 THROUGHPUT PREDICTION USING SIGNAL QUALITY

In this Section, we explore how the stochastic relationship between the throughput and the signal quality can be exploited to predict the achievable throughput without performing bandwidth measurements. For this, we first analyze how well we can approximate the properties of the empirical data, like the mean μ and the standard deviation σ using a mathematical model.

Since we know that the achievable throughput is different for different user equipment categories and cells with different bandwidths, we take this into consideration when creating our prediction.

In the following, we create a prediction model for the throughput that can be achieved using an LTE Cat 12 modem in 20 MHz cells. For this, we look at the throughput values obtained in 20 MHz cells and analyze them as a function of the measured SINR values. We aggregate the SINR values into groups of 10 dB to reduce noise in the data like we did in Section 3.1. For each group of throughput values, we calculate the mean μ and standard deviation σ. Finally, we try to approximate the μ and σ values using linear approximation.

The result of this evaluation can be seen in Fig. 4.1, where we compare the mean and standard deviation of the empirical throughput values grouped by their SINR value with their linear approximation. On the x-axis, we always show the middle value of a 10 dB group,i.e., a value of 10 dB means that the throughput value is the mean of the [5,15] dB group. Here, we can see that the linear approximation matches the empirical data reasonably well; however, not every detail can be described using this model. It is likely that with more measurement samples, the noise in the data would be reduced even further.

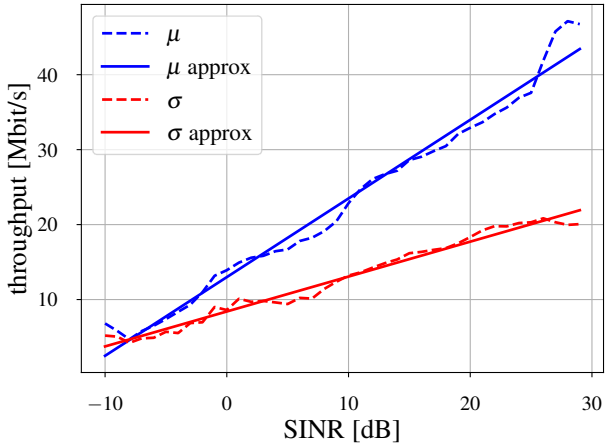
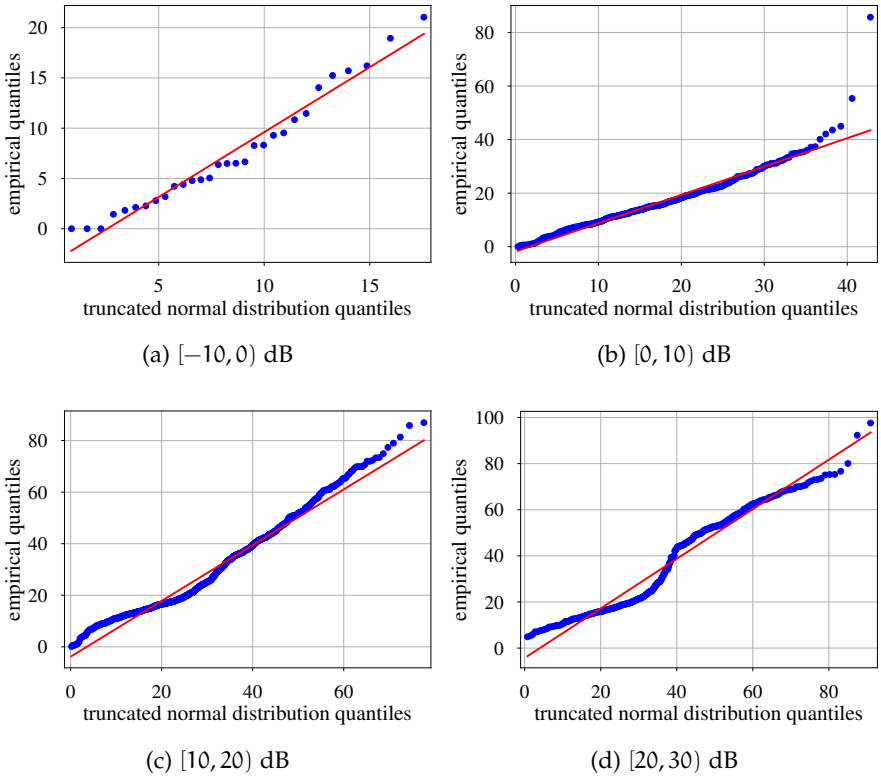


Figure 4.1: Comparison of empirical μ and σ of the throughput values with their linear approximation in 20 MHz cells.

Next, we model the distribution of the achievable throughput. For this, we take a truncated normal distribution and use the mean μ and the standard deviation σ we modeled in the previous step as parameters. We use a truncated normal distribution as we want to avoid the prediction of negative throughput values or values that are higher than the empirical values. To verify the result, we use Quantile-Quantile (Q-Q) plots to compare the quantiles of the empirical data and the modeled distribution.

As we can see in the resulting graphs in Fig. 4.2, a truncated normal distribution can model the distribution of the actual data reasonably well in many cases; however, it has its limitations. In Fig. 4.2b, we can see that some outliers at the edge of the distribution cannot



(a) $[-10, 0)$ dB (b) $[0, 10)$ dB

(c) $[10, 20)$ dB (d) $[20, 30)$ dB

Figure 4.2: Q-Q plots comparing the quantiles of the empirical data for different SINR ranges in 20 MHz cells.

be modeled very well. In Fig. 4.2c, we again see that the modeled distribution can be somewhat imprecise at the edges when compared to the empirical data. For the highest dB range depicted in In Fig. 4.2d, the modeled distribution tends to overestimate the throughput values in lower quantiles.

To visualize how well the distribution of the empirical throughput can be modeled with a truncated normal distribution, we also compare their CDFs in Fig. 4.3. Here, we can see that the modeled distribution matches the empirical data in many cases; however, it tends to overestimate the obtained throughput values in the 20-30 dB range.

Unfortunately, the small empirical values in the 20-30 dB range cannot be explained by signal quality values. These values may have been obtained while the cell load was higher than usual, and therefore the user equipment was scheduled less often. To improve the model, we would need to take the cell load into consideration. However, such data is not readily available to standard user equipment.

In conclusion, we can estimate the throughput distribution without performing throughput measurements by exploiting the stochastic relationship between the signal quality and the achievable throughput. Since the range of the throughput values in the resulting distribution

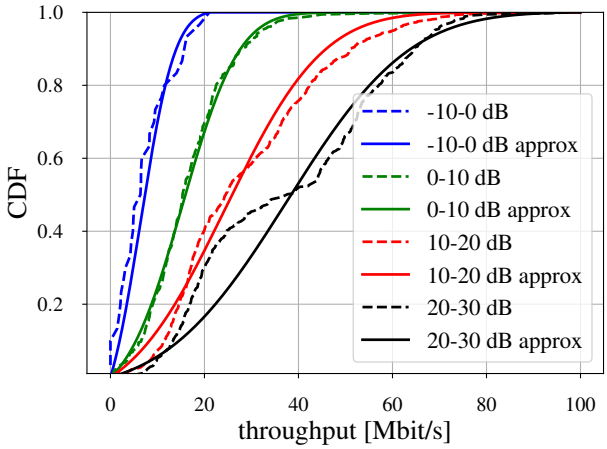


Figure 4.3: Comparison of the empirical throughput distributions for different SINR ranges with the modeled truncated normal distribution in 20 MHz cells.

can be relatively high, we can only use such a prediction to provide stochastic guarantees and not perform exact throughput predictions. Unfortunately, the effect of the cell load can be quite high and can significantly limit the obtained throughput even when the signal quality is high. This effect cannot be easily considered in such a model as cell load information is not available to the user equipment. We further discuss stochastic prediction and their use in ABR application in Chapter 5.

4.2 IMPACT OF THE DISTANCE BETWEEN MEASUREMENTS

The precision of a throughput prediction could further be improved by not only considering the frequency of the channel, its bandwidth, and the signal quality, but also the location of the measurement. In mobile scenarios, it is likely that when a user equipment downloads data at a certain location, the next download will occur at a neighboring location. In this Section, we evaluate the similarity of the throughput in neighboring locations.

In order to answer this question, we transform the (two-dimensional) GPS coordinates of routes depicted in Fig. 4.4 into one dimension, which we define as the distance to the reference point marked **x** (on the left) in Fig. 4.4a and 4.4b. Next, we divide the route into equidistant points (lags) and assign the closest throughput measurement to each lag. Then, we compute the autocorrelation of the downlink throughput for the defined transformation and different distances between measurement points, i.e., different lags. Autocorrelation refers to the correlation of the throughput values to values measured in the past, i.e., if the autocorrelation is high over multiple lags, the throughput values will be similar in neighboring locations. If the autocorrelation

(a) Highway route, where measurements have been performed in 800 MHz cells with 10 MHz bandwidth.

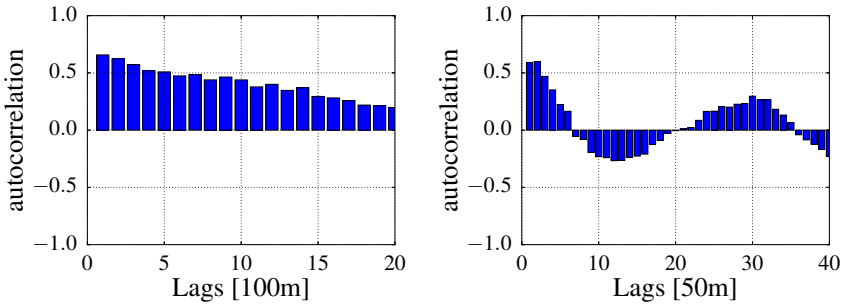
(b) City route, where measurements have been performed in 1800 MHz cells with 20 MHz bandwidth.

Figure 4.4: Highway and city routes where measurements have been performed. Map data © OpenStreetMap contributors, CC BY-SA [26].

is low, then the throughput values obtained in one location will not be able to be used to predict throughput at a neighboring location.

In Fig. 4.5a, we present the autocorrelation of downlink throughput for the rural route. In the first five lags (corresponding to a distance of 500 m), a high autocorrelation ($r > 0.5$) can be observed. Further, the autocorrelation decreases with the lags. In the city, the autocorrelation declines more quickly and is greater than 0.5 only for the first two lags, see Fig. 4.5b. This means that when a user equipment achieves some throughput at one location, it is likely to achieve a similar throughput in a 500 m range in a rural area but only a 100 m range in an urban area. This result also corresponds to what we have observed in the spatial distribution of the cells in Section 3.1: In the city, we have a more frequent occurrence of cells in the B3 band, which have a lower range than the cells in the B20 band, which mainly occur in rural areas. This means that in the city, it is more likely to switch the cell when changing the location. Different cells can experience different loads, which can heavily impact the available throughput. In addition, buildings, vehicles, and other obstacles occur more frequently in urban areas and can temporarily impact a UE's connection to an eNodeB.

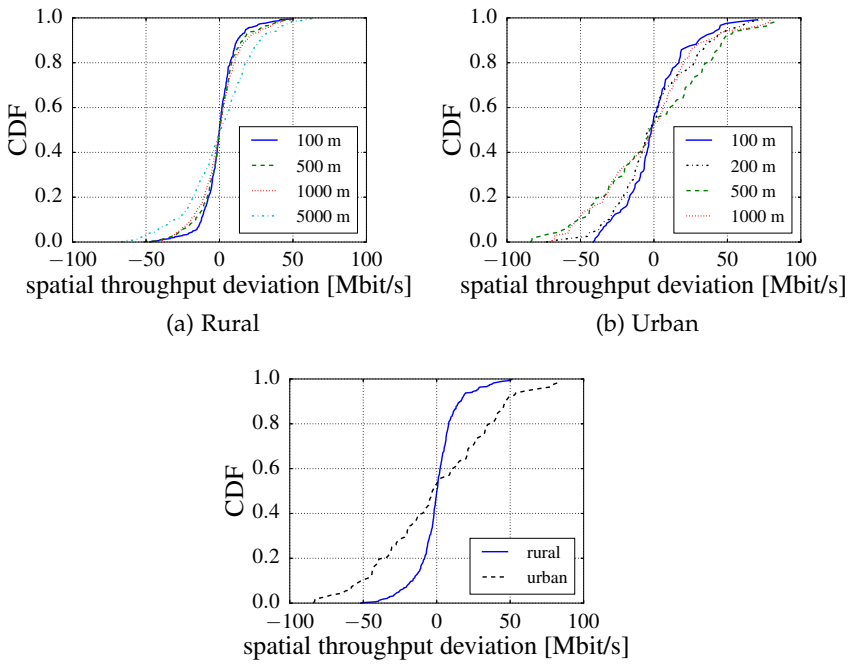
A more intuitive interpretation of this result can be obtained when observing the distribution of spatial throughput differences. For this, we look at every throughput measurement and compare it to the values obtained in a distance of 100 m, 200 m, etc. In Fig. 4.6, we present



(a) Rural

(b) Urban

Figure 4.5: Autocorrelation of download speed in rural and urban regions.



(a) Rural

(b) Urban

(c) Rural vs. Urban (500m lag)

Figure 4.6: Spatial variability of download speed in rural (10 MHz) and urban (20 MHz) regions.

the distribution of the differences of the throughput values based on their distance from each other. A value of 50 Mbit/s corresponds to a throughput change of this amount in, e.g., 100 m. If the value becomes negative, the throughput is reduced by 50 Mbit/s.

Fig. 4.6a shows that in rural areas, the probability of obtaining a significantly different throughput is small for a lag size up to 1000 m (relatively steep curves). For a distance of 5 km, we have a greater difference (the curve becomes more flat) which means that the probability of obtaining a substantially different throughput increases, i.e., the variability is higher. Fig. 4.6b shows the differences for the urban regions. Here, the throughput is stable only for the first 200 m, i.e., we have much higher spatial fluctuations. When we directly compare the throughput distribution for a specific lag (e.g., 500m in Fig. 4.6c), we can also see more directly that the throughput in rural areas experiences less change at a different location than in urban areas.

4.3 CONCLUSION

In this Chapter, we have analyzed how well the expected throughput of a 4G connection can be predicted using information about signal quality. For this, we have analyzed the measurements we performed using our LTE Cat 12 modem in cells with a 20 MHz bandwidth. Our analysis has shown that it is possible to model the obtainable through-

put with a truncated normal distribution. However, such a model cannot consider load-based effects, meaning that the instantaneous throughput can be lower than predicted by the model if the cell load becomes high.

We have also analyzed the relationship between throughput measurements at different locations. Our analysis has shown that in rural areas, the obtained throughput can remain stable over distances of up to 1000 meters. In contrast, the throughput in urban areas can fluctuate over a much shorter distance of 100 meters. This means that especially in urban areas, throughput predictions must be performed more frequently to obtain a more reliable estimate.

5 IMPROVEMENT OF DEPENDABILITY

In the previous Chapter, we discussed the possibility of making predictions of achievable data rates in live LTE networks. We have concluded that stochastic predictions can be made based on the measured signal quality. We have also observed that when the user equipment is mobile, the achievable data rates experience only small changes in rural areas. In contrast, these changes are much more significant in urban areas.

In this Chapter, which is based on [57] and includes additional explanations, we will discuss how stochastic predictions can be used to improve the behavior of mobile applications. For this, we use the example of chunk-based media streaming. Our results show that when information about achievable data rates is available, it is possible to limit the probability of video freezes to a desired upper bound. The trade-off is that when this upper bound is selected conservatively, the video quality selection is also very conservative.

5.1 ADAPTIVE BITRATE STREAMING WITH PERFORMANCE PREDICTION

In recent years, online video streaming has become one of the most important Internet applications, accounting for 71% of all mobile Internet traffic as of November 2022 [58]. Hence, it is important to adapt video playback methods to the challenges of mobile environments that have a highly volatile throughput due to the variability of the wireless channel. We have already discussed the extent of this variability in LTE in Chapters 3 and 4.

Such volatile network behavior can be challenging for applications that require a constant flow of data, like video streaming applications. ABR streaming algorithms try to solve this problem by adjusting the bitrate of the application according to a predefined metric. Typically, such an algorithm would perform passive bandwidth estimation at the receiver and adjust the sending rate of the data based on the result.

A commonly used video streaming method is for the server to divide the video into chunks of fixed duration and for the client to download these chunks one after another. The server typically offers these chunks to the client at different bitrates, i.e., different video coding quality. This is also visualized in Fig. 5.1. After downloading each chunk, the client estimates the currently available throughput and selects the bitrate of the next chunk based thereon. Examples of such adaptive bitrate streaming methods include Apple's HTTP Live

Streaming (HLS) [59] and Moving Picture Experts Group - Dynamic Adaptive Streaming over HTTP (MPEG-DASH) [60].

Figure 5.1: ABR video streaming applications like MPEG-DASH typically divide a video into chunks and make each chunk available in different quality levels.

The client typically maintains a de-jitter buffer where it can store several chunks before playback starts. This pre-buffering can, within certain limits, cope with possible fluctuations of the network conditions. In our work, we discuss how to adapt the bitrate of a video stream in order to avoid buffer underflow.

Methods for bitrate adaptation can be roughly divided into two groups: throughput-based and buffer-based. Throughput-based methods like ELASTIC [61] or PANDA [62] often use an estimate of the throughput based on the average performance in the past. In contrast, buffer-based methods like BOLA [63] consider the current level of the de-jitter buffer at the client (and possibly the throughput in addition) [64].

Existing strategies often try to achieve multiple goals: (i) They want to stream the video in a quality that is as high as possible, (ii) they want to minimize video freezes, and (iii) they want to reduce the number of quality switches. Often the first goal conflicts with the other two, meaning that a balance between the goals has to be found.

In mobile scenarios, throughput-based strategies can exhibit poor performance since their knowledge of available throughput in the past does not necessarily inform them about the throughput they can achieve in the future. This is because the network capacity can be different when the location of the user equipment changes. When using cellular communication like 4G or 5G, instead of using the information about the throughput achieved in the past, it is possible to use a prediction of achievable throughput based on current signal quality information. When information about the distribution of the achievable throughput is available (e.g., based on the signal quality or past performance), it is also possible to provide stochastic guarantees for the buffer of an ABR streaming application not to run empty.

In this Chapter, we propose a buffer-based adaptive bitrate streaming method that accounts for the variability of throughput by not only using an estimate of the average throughput but by using an estimate of the throughput distribution. A way to obtain such a throughput distribution for mobile networks is described in Chapter 4.

Our method aims to provide the highest possible video bitrate while not exceeding a defined buffer underflow probability. We use stochastic network calculus to derive the maximum bitrate for the general case of any throughput distribution.

In the second step, we use our model from Chapter 4 to provide analytical and simulation results for the specific example of the throughput having a stationary Gaussian distribution. We have chosen this distribution as it has been a good model to represent our empirical measurement results in most cases. However, our method also works with models more commonly used to describe the behavior of wireless channels, such as Rayleigh fading [65].

An analysis of a related problem using similar mathematical methods is presented in [66]. Different from our work, the authors analyze the fill level of a send buffer that stores data for transmission to an LTE base station. Here, the goal is to adapt the transmission resources to prevent the send buffer from overflowing instead of, in our case, adapting the source data rate to avoid underflow of the de-jitter buffer at the receiver.

Another related problem is discussed in [67]. Here, the authors analyze a wireless video streaming scenario for which they provide a probabilistic lower bound on the received video quality and use this result for transmission rate adaptation. Different from our work, the authors assume a scenario with strict delay constraints so that the variability of the wireless channel cannot be compensated by a de-jitter buffer at the receiver.

The main contribution of this work is a model of a de-jitter buffer for video playback that can be used to calculate the probability of buffer underflow. We also show how to calculate the maximum video coding bitrate for which a desired buffer underflow probability is not exceeded. Additionally, we provide simulation results for the concrete example of a Gaussian throughput distribution. Our simulation results show that a system that adjusts the video bitrate according to our method does not experience buffer underflow within the limits of the specified probability.

5.2 BUFFER MODEL

We analyze a model of a video streaming application that maintains a de-jitter buffer in which all downloaded data is stored before being played back. We denote the download process, i.e., the arrivals to the buffer, in the time interval $[\tau, t)$ by $A(\tau, t)$. When the data is played

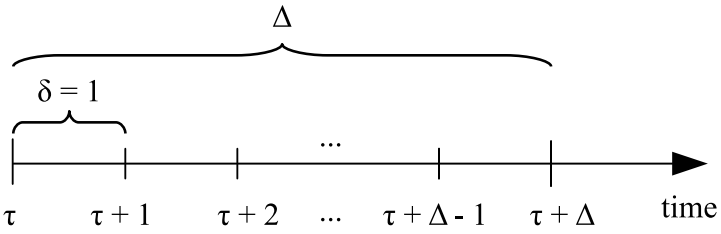


Figure 5.2: Time interval $[\tau, \tau + \Delta)$ with granularity $\delta = 1$.

back, it is removed from the buffer. For a time interval that starts at time τ and has a duration Δ (e.g., a video chunk), we want to calculate the probability that the buffer level is insufficient, causing an interruption in playback, i.e., buffer underflow. Additionally, we want to calculate the probability that the buffer level stays above some safety margin β after each interval. Our model uses discrete time with time-slot duration δ, see Fig. 5.2. For notational simplicity, we normalize $\delta = 1$.

In order to determine if a video can be played back without interruption in a time interval $[\tau, \tau + \Delta)$ we have to be able to determine the buffer level at any given time $t \in [\tau, \tau + \Delta)$. This is because we must ensure that the buffer does not underflow at any point. Different from the usual convention, we define our buffer level in units of playback time instead of in units of bits. For a given video bitrate r, the scaling is achieved by dividing the number of bits in the buffer by r. Similarly, the arrivals to the buffer and the playback process from the buffer are scaled by division by r. As a consequence, the playback process in the interval $[\tau, t)$ is simply $t - \tau$, regardless of the video bitrate. We use this scaling as it avoids difficulties in the formulation of adaptive bitrate streaming that would otherwise occur, since the buffer may contain parts of different video chunks that are encoded at different bitrates. A general theory of non-constant scaling processes is provided in [68] and the following works. Using the scaling by r, we can calculate the buffer level at time $t \geqslant \tau \geqslant 0$ as

$$B(t) \geqslant B(\tau) + \frac{A(\tau, t)}{r} - (t - \tau), \tag{5.1}$$

where $B(t)$ is the buffer level at time t and $B(\tau)$ is the buffer level that was available at time τ. Both $B(t)$ and $B(\tau)$ are measured in units of δ. Eq. (5.1) is an exact equality if no buffer underflow occurs in $[\tau, t)$, i.e., if the playback is never interrupted. A visual representation of the buffer process can be seen in Fig. 5.3.

Due to the random nature of wireless channels, the cumulative arrivals to the de-jitter buffer $A(\tau, t)$ in $[\tau, t)$ are random. In each time slot $[\tau, \tau + 1)$ for $\tau \geqslant 0$, a random amount of data $X(\tau)$ will be downloaded. We assume that the increments $X(\tau)$ are independent and identically distributed (iid). Each increment has the Moment-

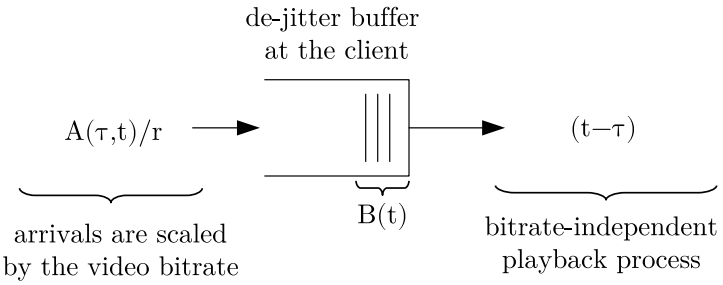


Figure 5.3: In the interval $[\tau, t)$ the buffer level $B(t)$ is increased by arrivals $A(\tau, t)$ that have been scaled by the video bitrate r and decreased by the elapsed play back time $(t - \tau)$.

Generating Function (MGF) $M_X(\theta) = E[e^{\theta X}]$, $\theta \in \mathbb{R}$. We can calculate the cumulative arrivals $A(\tau, t)$ for the interval $[\tau, t)$ as

$$A(\tau, t) = \sum_{i=\tau}^{t-1} X(i).$$

The arrival process is visualized in Fig. 5.4.

Since the cumulative arrivals $A(\tau, t)$ are random, so is the buffer level $B(t)$. In order to be able to play back a video in an interval $[\tau, t)$ without interruption, the buffer has to stay above a certain threshold b_{min} in every single time step. The value b_{min} can be, e.g., 0 or the size of a single frame. In the following, we want to calculate the probability of a buffer underflow, i.e., the probability of the buffer level falling to or below b_{min}.

Theorem 1 (Buffer Underflow Probability) *Consider a buffer whose fill level is governed by Eq. (5.1) with iid arrivals that have an MGF $M_X(\theta)$. An upper bound on the probability ε of buffer underflow in the time interval $[\tau, \tau + \Delta]$ is*

$$P\left[\exists t \in [\tau, \tau + \Delta) : B(t) \leqslant b_{min}\right] \leqslant e^{-\theta(B(\tau) - b_{min})} := \varepsilon$$

if $\theta > 0$ satisfies the condition

$$M_X\left(\frac{-\theta}{r}\right) e^{\theta} = 1.$$

Proof 1 *A buffer underflow in an interval $[\tau, \tau + \Delta)$ occurs when there exists a time step in which the buffer level is at or below the value b_{min}. Given the interval starts with an initial buffer filling of $B(\tau)$, the probability of this happening follows with Eq. (5.1) as*

$$\begin{aligned}
&P[\exists t \in [\tau, \tau + \Delta) : B(t) \leqslant b_{min}] \\
&\leqslant P\left[\min_{t \in [\tau, \tau+\Delta)} \left\{ B(\tau) + \frac{A(\tau, t)}{r} - (t - \tau) \right\} \leqslant b_{min}\right] \\
&= P\left[\max_{t \in [\tau, \tau+\Delta)} \left\{ e^{\theta\left((t-\tau) - \sum_{i=\tau}^{t-1} X(i) r^{-1}\right)} \right\} \geqslant e^{\theta(B(\tau) - b_{min})}\right]. \quad (5.2)
\end{aligned}$$

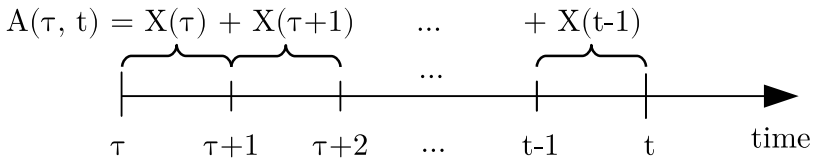


Figure 5.4: In each time step a random amount of data $X(i)$ is downloaded. The sum of all downloaded data is $A(\tau, t)$.

We want to find an upper bound on this probability by using Doob's martingale inequality [69]. The inequality is frequently used in the stochastic network calculus, e.g., [66, 70, 71, 72, 73, 74]. For this, we first need to find a θ *for which the process*

$$U(t) = e^{\theta\left((t-\tau)-\sum_{i=\tau}^{t-1} X(i) r^{-1}\right)} \tag{5.3}$$

becomes a martingale, i.e., we need to show that

$$E[U(t+1)|U(t), U(t-1), ..., U(\tau)] = U(t). \tag{5.4}$$

From Eq. (5.3) we get $U(t+1) = U(t)e^{\theta\left(1-X(t)r^{-1}\right)}$*, which has the conditional expectation*

$$\begin{aligned} &E[U(t+1)|U(t), U(t-1), ..., U(\tau)] \\ &= U(t) M_X\left(\frac{-\theta}{r}\right) e^{\theta}, \end{aligned}$$

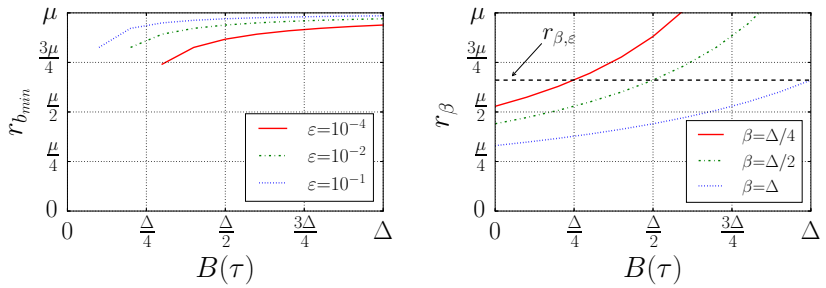
where we used the independence of the increments $X(i)$*. Clearly,* $U(t)$ *satisfies the condition from Eq. (5.4) when* θ *satisfies the constraint from Th. 1.*

We can now use a version of Doob's martingale inequality [71, Lem. 2] $P[\max_{t\in[\tau,\tau+\Delta)} U(t) \geqslant x] \leqslant E[U(\tau)]x^{-1}$ *for non-negative* $U(t)$ *and* $x > 0$*, where we shifted the origin of the sequence* $U(t)$ *from 1 to* τ*. Application of Doob's inequality to the martingale in Eq.* (5.2) *where* $x = e^{\theta(B(\tau)-b_{min})}$ *and* $E[U(\tau)] = 1$ *completes the proof.*

The probability ε from Th. 1 can be calculated for general arrivals $A(\tau, t)$ with iid increments $X(i)$ that have the MGF $M_X(\theta)$. In the following example, we show how the probability from Th. 1 can be calculated for a Gaussian distribution. We use this distribution as we have observed in Chapter 4 that it can be used to model our empirical measurement results of achievable throughput.

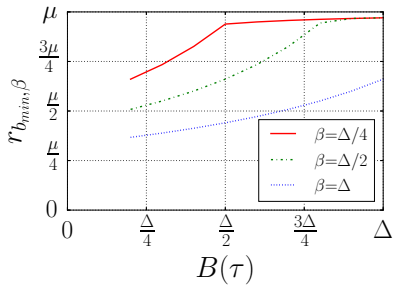
Example 1 *In the following, we consider a simple example of a channel that offers a mean throughput* μ *that varies over time with a variance* σ^2*. We assume that the throughput of this channel has a Gaussian distribution with* $M_X(\theta) = e^{\left(\mu\theta+\frac{\sigma^2\theta^2}{2}\right)}$*. Using this MGF, we can obtain the value* $\theta = 2r(\mu - r)\sigma^{-2}$ *for which the constraint of Th. 1 is satisfied. By inserting this value into the definition of* ε*, we obtain the probability of the buffer falling to or below the minimum threshold* b_{min} *as*

$$\varepsilon = e^{-2r(\mu-r)\sigma^{-2}(B(\tau)-b_{min})}. \tag{5.5}$$

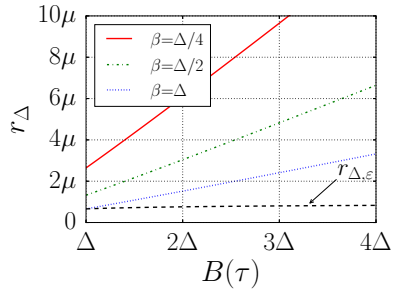


(a) $r_{b_{min}}$ is the maximum rate allowed so that the buffer does not underflow at any time step.

(b) r_β is the maximum rate allowed so that a safety margin β is achieved at time $\tau + \Delta$. $r_{\beta,\varepsilon}$ is the minimal rate with probability $1 - \varepsilon$.



(c) $r_{b_{min},\beta}$ is the minimum of the two rates $r_{b_{min}}$ and r_β.



(d) $r_\Delta$is the maximum rate allowed so that the a safety margin β is achieved at time $\tau + n\Delta$. $r_{\Delta,\varepsilon}$ is the minimal rate with probability $1 - \varepsilon$.

Figure 5.5: Graphs of the rates $r_{b_{min}}$, r_β, $r_{b_{min},\beta}$ and r_Δ for parameters $X(i) \sim \mathcal{N}(4, 2)$, $\Delta = 10$, $b_{min} = 0, \varepsilon = 10^{-2}$.

Bitrate Adaptation for b_{min}: In Th. 1, we have shown how to calculate a probability of buffer underflow in a time interval $[\tau, t)$ given iid arrivals $A(\tau, t)$ that are scaled by a bitrate r. We now want to consider the case where we have some desired upper bound on the buffer underflow probability ε, for which we want to calculate the maximum bitrate of the video that does not violate the bound. In the following, we call this rate $r_{b_{min}}$, as it is the maximum source bitrate for which the threshold b_{min} is not violated with probability ε. This bitrate can be calculated at the beginning of each interval $[\tau, \tau + \Delta)$ to adjust the video stream to the maximum quality for which a buffer underflow is unlikely. Note that the rate $r_{b_{min}}$ can be larger or smaller than the bitrate from the previous interval.

Generally, we can obtain the bitrate $r_{b_{min}}$ for a given arrival MGF by finding a θ that satisfies the condition from Th. 1, inserting this θ into the definition of ε and solving for r. Note that the rate $r_{b_{min}}$ obtained this way is a positive real number. In systems where the rate can only be adjusted in discrete steps, as is the case with, e.g., MPEG-DASH, the

bitrate can be chosen as the largest possible bitrate that is lower than $r_{b_{min}}$.

Example 2 *For the special case of Gaussian increments that has been introduced in Example 1, we can now obtain the value $r_{b_{min}}$ by solving Eq. (5.5) for r. To keep the example simple, we choose $b_{min} = 0$. We then get*

$$r_{b_{min}} = \frac{\mu}{2} + \sqrt{\frac{\mu^2 B(\tau) + 2\ln(\varepsilon)\sigma^2}{4B(\tau)}}. \tag{5.6}$$

Using this example, we now discuss how $r_{b_{min}}$ is affected by the available buffer level $B(\tau)$ and the desired violation probability ε. A graph showing the influence of these parameters on the rate $r_{b_{min}}$ can be seen in Fig. 5.5a. The x-axis is scaled to the interval length Δ, and the y-axis is scaled to the mean download rate μ to emphasize the influence of these parameters on the source bitrate. For small values of $B(\tau)$, the violation probability ε has more influence on the rate $r_{b_{min}}$ than for larger values. For very small values of $B(\tau)$ no rate $r_{b_{min}}$ can be calculated. This is because a certain amount of pre-buffering is required for the system to function correctly regardless of the rate. Note that the amount of required pre-buffering increases when ε becomes more conservative. For any choice of ε the suggested bitrate $r_{b_{min}}$ approaches the mean throughput μ as the buffer level $B(\tau)$ increases. The rate never exceeds μ as that would violate the condition $\theta > 0$.

Safety Margin: As we have seen in Eq. (5.6) and Fig. 5.5a, some pre-buffering is required to achieve a desired violation probability ε - e.g. it should be about $\frac{\Delta}{4}$ for $\Delta = 10$ and $\varepsilon = 10^{-2}$. Besides, an additional safety margin might be desirable to avoid application failure if the average download rate changes rapidly. This safety margin can be reached by ensuring that a certain target buffer level $\beta > 0$ is available at the end of a time interval $[\tau, \tau + \Delta)$. We can generalize this for longer intervals of length $n\Delta$, $n \geqslant 1$ by guaranteeing this safety margin at the end of a longer time interval $[\tau, \tau + n\Delta)$, i.e., $B(\tau + n\Delta) \geqslant \beta$. Note that since $B(\tau)$ is a scaled buffer in units of δ, the safety margin β is also in units of δ. In the following, we want to calculate the probability of violating the safety margin β at the end of an interval.

Theorem 2 (Probability of Violating the Safety Margin β) *Consider a buffer whose fill level is governed by Eq. (5.1) with iid arrivals that have an MGF $M_X(\theta)$. An upper bound on the probability ε of violating the safety margin β at the end of the interval $[\tau, \tau + n\Delta)$ is*

$$\begin{aligned} &P[B(\tau + n\Delta) \leqslant \beta] \\ &\leqslant \min_{\theta \geqslant 0} \left\{ e^{\theta r(\beta + n\Delta - B(\tau))} M_X(-\theta)^{n\Delta} \right\} := \varepsilon. \end{aligned}$$

Proof 2 *In order to calculate the probability of violating the safety margin β at the end of the time interval $[\tau, \tau + n\Delta)$ it is enough to only observe the*

buffer level at the end of the interval, i.e., at time $t = \tau + n\Delta$*. The probability of violating the safety margin* β *follows with Eq. (5.1) as*

$$\begin{aligned} &P[B(\tau + n\Delta) \leqslant \beta] \\ &\leqslant P[A(\tau, \tau + n\Delta) \leqslant r(\beta + n\Delta - B(\tau))]. \end{aligned}$$

By applying Chernoff's bound $P[X \leqslant x] \leqslant e^{\theta x} E\left[e^{-\theta X}\right]$ *for* $\theta \geqslant 0$ *and using the independence of the increments* $X(i)$ *we directly obtain the probability in Th. 2.*

The probability in Th. 2 can be calculated for arbitrary arrivals $A(\tau, \tau + n\Delta)$ with iid increments $X(i)$ which have the MGF $M_X(\theta)$. If we select a specific distribution of the increments, we can calculate the value θ for which this probability becomes minimal.

Example 3 *For Gaussian increments, which we defined in Example 1, the probability from Th. 2 is minimal for*

$$\theta = \frac{n\Delta\mu - r(\beta + n\Delta - B(\tau))}{n\Delta\sigma^2}$$

By inserting this value of θ *into the definition of* ε *we get*

$$\varepsilon = \exp\left(\frac{-(n\Delta\mu - r(\beta + n\Delta - B(\tau)))^2}{2n\Delta\sigma^2}\right)$$

as the probability for violating the safety margin β *at the end of the interval* $[\tau, \tau + n\Delta)$.

Bitrate Adaptation for β in $[\tau, \tau + \Delta)$: After obtaining the probability for violating the safety margin β, we can calculate r_β, which is the maximal video bitrate that ensures that the buffer level is at least β at the end of the interval $[\tau, \tau + \Delta)$ with a violation probability ε.

Example 4 *Using the probability from Example 3 for* $n = 1$ *and* $B(\tau) < \Delta$ *we can calculate*

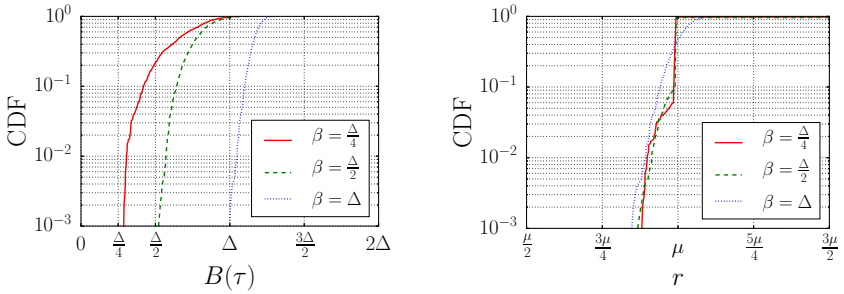
$$r_\beta = \frac{\Delta\mu - \sqrt{-2\Delta \ln(\varepsilon)\sigma^2}}{\beta + \Delta - B(\tau)}. \tag{5.7}$$

We can also calculate a minimum source bitrate $r_{\beta,\varepsilon}$ from Eq. (5.7) that applies if $B(\tau)$ equals β. Since $B(\tau) \geqslant \beta$ with probability $1 - \varepsilon$, the rate r_β falls below $r_{\beta,\varepsilon}$ at most with probability ε. In Fig. 5.5b, we present a graph of the rate r_β for different values of β. It can be seen that r_β grows with increasing $B(\tau)$ without being constrained by μ as was the case for $r_{b_{min}}$. This is because, different from $r_{b_{min}}$, a specific level of the buffer only has to be guaranteed at the end of the large interval $[\tau, \tau + \Delta)$ and not at every single time step, therefore as $B(\tau)$ increases, the probability of not having a buffer level β at time $\tau + \Delta$ approaches 0. The black dashed line indicates the minimum rate $r_{\beta,\varepsilon}$.

Note that $r_{\beta,\varepsilon}$ is independent of β, meaning the minimum buffer level does not affect the minimum source bitrate. The parameter β only influences how much the source bitrate changes with increasing $B(\tau)$, the changes being higher for less conservative values of β.

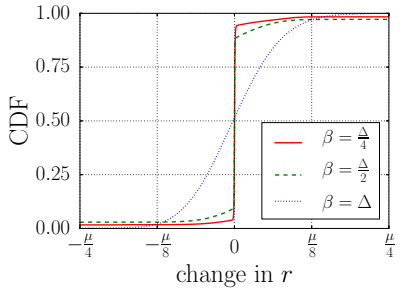
In the period $[\tau, \tau + \Delta)$ both constraints b_{min} and β have to be satisfied. For this, we have to select the minimum of the two bitrates:

$$r_{b_{min},\beta} = \min\{r_{b_{min}}, r_\beta\}. \tag{5.8}$$



(a) Stationary distribution of $B(\tau)$

(b) Stationary distribution of r



(c) Variability of r between intervals

Figure 5.6: Effect of different values of the safety margin β on the buffer level $B(\tau)$ and the source bitrate r for $X(i) \sim \mathcal{N}(4, 2)$, $\Delta = 50$, $b_{min} = 0, \varepsilon = 10^{-2}$ and $\beta \in \left\{\frac{\Delta}{4}, \frac{\Delta}{2}, \Delta\right\}$.

In Fig. 5.5c, we present a graph of the source bitrate $r_{b_{min},\beta}$ for different values of β and $\varepsilon = 10^{-2}$. It can be seen that for smaller values of $B(\tau)$, the rate r_β is the minimum of the two rates, whereas for larger values, the minimum is generally $r_{b_{min}}$. The switching point between the two rates depends on the choice of β: For less conservative values, the switch happens earlier than for the more conservative ones (if at all) since the source bitrate is adjusted towards the average throughput more quickly.

Bitrate Adaptation for β in $[\tau, \tau + n\Delta)$: In the case that the buffered data is sufficient for the upcoming interval $[\tau, \tau + \Delta)$, meaning $B(\tau) \geqslant \Delta$, we no longer have to adjust the source bitrate to ensure that the buffer does not fall below the threshold b_{min} at some time t in $[\tau, \tau + \Delta)$. This has the advantage that we can select a source bitrate

that is no longer bounded by the mean download rate μ as has been the case with $r_{b_{min}}$ in Fig. 5.5a. Instead, we can now choose a new source bitrate r_Δ in a way that ensures that the safety margin β is achieved after a larger interval $[\tau, \tau + n\Delta)$ with probability $1-\varepsilon$, where $n = \frac{B(\tau)}{\Delta}$, $n \in \mathbb{R}$.

Example 5 *Similar to Example 3, for Gaussian increments we can find a θ for which ε from Th. 2 is minimal and solve for r_Δ in order to obtain the maximum bitrate for which $B(\tau + n\Delta) \geqslant \beta$ with violation probability ε:*

$$r_\Delta = \frac{n\Delta\mu - \sqrt{-2n\Delta \ln(\varepsilon)\sigma^2}}{\beta + n\Delta - B(\tau)}. \tag{5.9}$$

Again, we can calculate a minimum bitrate $r_{\Delta,\varepsilon}$ from Eq. (5.9) that applies if $B(\tau)$ equals β. Since $B(\tau) \geqslant \beta$ with probability $1-\varepsilon$, the rate r_Δ falls below $r_{\Delta,\varepsilon}$ at most with probability ε. The resulting graph of r_Δ for different values of β and $\varepsilon = 10^{-2}$ is shown in Fig. 5.5d. The bitrate grows linearly with the available buffer $B(\tau)$. The slope of the rate is smaller for more conservative choices of β. Because the buffer level $B(\tau)$ is very large, the resulting source bitrate is almost always larger than μ. The black dashed line indicates the minimum source bitrate $r_{\Delta,\varepsilon}$. This rate is independent of β; however, it does grow very slightly with $B(\tau)$.

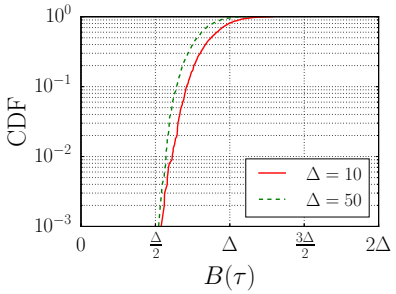
Finally, for all values of $B(\tau)$ the source bitrate selection can be written as

$$r = \begin{cases} r_{b_{min},\beta} & \text{, for } B(\tau) < \Delta \\ r_\Delta & \text{, for } B(\tau) \geqslant \Delta. \end{cases} \tag{5.10}$$

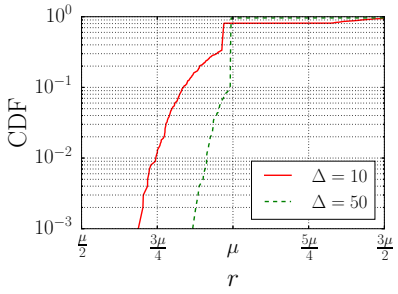
5.3 SIMULATION RESULTS

In this Section, we present and discuss simulation results to illustrate the performance of the system that adapts its bitrate according to the method described in Eq. (5.10). For this, we have written a discrete-time simulator. In each time step δ the buffer level is increased by a Gaussian increment $X(i) \sim \mathcal{N}(\mu, \sigma^2)$ that is scaled by the current source bitrate r, and decreased by δ. Each simulation run is configured to consist of 10^3 intervals Δ where $\Delta = 50\delta$.

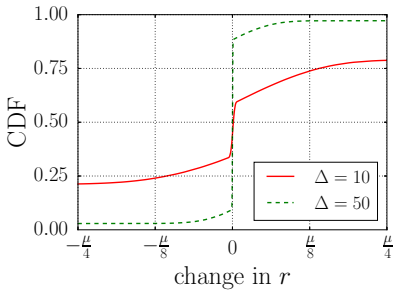
In all performed simulation runs, both the source bitrate r and the buffer level $B(\tau)$ over time converged to a stationary distribution within the first few intervals Δ. The shape of these distributions depends on the parameters $\mu, \sigma^2, b_{min}, \beta, \Delta$, and ε. In the following, we will discuss how some of these parameters affect the resulting distribution by presenting the corresponding simulation results. For



(a) Stationary distribution of $B(\tau)$



(b) Stationary distribution of r



(c) Variability of r between intervals

Figure 5.7: Effect of different values of the interval length Δ on the buffer $B(\tau)$ and the source bitrate r for $X(i) \sim \mathcal{N}(4, 2)$, $b_{min} = 0, \varepsilon = 10^{-2}$, $\beta = \frac{\Delta}{2}$ and $\Delta \in \{10, 50\}$.

each set of parameters, we perform 10^3 simulation runs and show the distribution of the results.

First, we discuss the effect of β on the resulting distributions of the buffer level $B(\tau)$ and the source bitrate r. In Fig. 5.6a, it can be seen that with increasing target level β, the distribution of the buffer level also becomes larger and less variable. Note that for the selected probability $\varepsilon = 10^{-2}$ the buffer level $B(\tau)$ satisfies both specified constraints β and b_{min}.

In Fig. 5.6b, we can see that for small values of β, the source bitrate stays stable in about 90% of the cases and that there are not many differences in the distributions based on β. For $\beta = \Delta$, the distribution becomes more variable; however, the median remains the same. The variability is small for buffer levels $B(\tau) < \Delta$ because the source bitrate is selected according to Eq. (5.8) as shown in Fig. 5.5c, making it close to the mean download rate μ and changing only very slightly with changes in $B(\tau)$. For $B(\tau) \geqslant \Delta$, the source bitrate is selected as shown in Fig. 5.5d, meaning it increases and decreases linearly with any change in $B(\tau)$, resulting in higher variability. We further investigate the effect of the parameter β on the variability of the source bitrate in Fig. 5.6c by plotting the distribution of the changes between neighboring intervals. It can be seen that for smaller values of β, only slight changes of the source bitrate should be expected when

going from one interval to the next, with large changes being very rare. For $\beta = \Delta$, the source bitrate changes with every new interval; however, it stays the same on average. This occurs because of the linear behavior of r_Δ that can be seen in Fig. 5.5d. A more conservative rate selection method for large safety margins $\beta \geqslant \Delta$ may remedy these rate fluctuations.

The interval length Δ also affects the distributions of $B(\tau)$ and r. The comparison of the buffers in relation to Δ in Fig. 5.7a tells us that for higher values of Δ, the buffer is slightly smaller, which is due to the median source bitrate being higher, as can be seen in Fig. 5.7b. Here we can also see that for the higher value of Δ, the source bitrate r remains stable with a higher probability, whereas for the smaller Δ, the source bitrate is more variable. The influence of Δ on rate changes between intervals can be seen in Fig. 5.7c: For the larger Δ, the probability of a rate change is very small, whereas for a smaller Δ, this probability increases. Overall, it can be said that choosing a larger Δ yields a more stable distribution of bitrates as the variability of the individual time steps becomes less important over longer intervals. Note that increasing Δ also means that the bitrate can be adapted less frequently.

5.4 CONCLUSION

In this Chapter, we have presented and analyzed a buffer-based bitrate adaptation technique for variable bitrate video streaming. We have derived a buffer underflow probability for the general case of the arrivals having any iid distribution and shown how this probability can be used for bitrate selection using an example of a specific distribution. Additionally, we have calculated a minimum possible bitrate that is only violated with a small probability when the rate is adjusted according to our method. Using a series of simulations, we have analyzed the impact of different system parameters on the selected bitrate. In all performed simulation runs, regardless of the initial buffer level, both the bitrate and the buffer level converged to stationary distributions after only a few intervals. For all selected parameters, the simulated system did not violate the specified buffer underflow probability. We have also observed that the bitrate distribution becomes more stable and that the minimum bitrate becomes larger when the interval length Δ increases, i.e., when adaptation occurs less frequently. We have shown that our system requires some minimum buffer level which depends on the desired maximal probability of buffer underflow, i.e., video stalling. Due to how our method adjusts the bitrate, the bitrate distribution becomes more variable with an increase of the desired safety margin. This might be different for more conservative bitrate selection methods.

6 CONCLUSIONS AND FUTURE WORK

6.1 CONCLUSIONS

In this thesis, we have discussed the challenges faced when using LTE for dependable mobile communication in vehicular scenarios. Our approach consisted of

- Evaluating the performance of LTE and how it has developed over time, including specific mechanisms like carrier aggregation.
- Analyzing how the performance of applications run over LTE can be affected by the selection of different transport layer congestion control protocols.
- Performing measurements to analyze the relationship between the measured signal quality and obtained throughput.
- Predicting the performance of LTE using information about the signal quality.
- Utilizing the prediction of the achievable data rates to improve the outage performance of applications with adjustable data usage.

Our analysis of LTE's performance has shown that it can be quite volatile for mobile users: As the user position and distance relative to the base station are correlated with the capacity of the channel, both the achievable throughput and the delays can experience significant fluctuations when the user equipment changes its position. The user's location also has a high impact on the performance: In our measurements, we have observed that there is often less infrastructure in rural than in urban areas, meaning that the performance of mobile communication can be reduced on highways.

When comparing the past and present performance of LTE, we have observed that it has improved with later versions due to the use of higher-order modulation schemes and carrier aggregation for combining the fragmented spectrum. However, due to limited infrastructure, especially in rural areas, it is not always possible to achieve higher performance with more recent LTE user equipment.

When looking at the performance of different congestion control algorithms, we have observed that some of the assumptions made by many of the existing algorithms are not always valid in cellular systems. In our measurements, TCP connections over LTE have rarely experienced packet loss, leading to oversized congestion windows and

very high delays when using loss-based algorithms like Reno or CUBIC. The delay-based algorithm Vegas stops increasing its congestion window very early as its baseRTT measurement looks for the lowest RTT, which can be much lower than the average value due to scheduling effects in LTE. In our measurements, TCP BBR has achieved the best performance: It has developed a congestion window that was able to utilize the entire available capacity without causing bufferbloat. It also changed the size of the congestion window when the capacity of the channel changed due to the user equipment changing its position. However, since it has been shown that BBR can be unfair towards other congestion control algorithms, the choice of the congestion control algorithm for LTE can still be improved.

Our measurements of the relationship between the signal quality and the achievable data rates have shown that while better signal quality is correlated with higher data rates, the relationship between the two is stochastic, i.e., better signal quality improves the probability for higher data rates but does not guarantee them.

Knowing the factors that affect the performance of LTE connections, we have proposed a model that predicts the achievable throughput based on signal quality measurements when given information about the user equipment capabilities and total channel bandwidth. Since our measurements have shown that the relationship between signal quality and achievable data rates is stochastic, the output of our method is a distribution of achievable throughput values. Our method uses a truncated normal distribution as it was well suited to model the empirical data that we have obtained in our measurements.

When a distribution of achievable data rates is available, it is possible to use this information to improve the outage probability of network applications with adjustable data rate usage, e.g., video streaming with MPEG-DASH. We have proposed a method that can limit the outage probability to a desired upper bound and verified it using a simulation. While limiting the outage probability works as intended, the trade-off is that the changes between different video quality levels can be abrupt and thus detrimental to the user experience.

Overall, it can be said that there is still much that can be improved when it comes to the dependability of mobile applications that use LTE. Even though the LTE specification has continuously increased achievable data rates, the actual expansion of the infrastructure was not always as fast. Especially in rural areas, the performance of LTE has not always improved throughout the years. There is also still room for improvement when it comes to the choice of the congestion control algorithm in LTE, even though BBR already delivers a very formidable performance. Finally, a lot can still be done in the applications themselves: Since volatile performance is to be expected, applications can use this knowledge to improve their dependability. Our proposed al-

gorithm for the rate selection in MPEG-DASH is only a small example of how the dependability of mobile applications can be improved.

6.2 FUTURE WORK

The development of the cellular network has already reached the next step with the release of 5G NR. To guarantee dependable performance of mobile applications in 5G networks, it is also necessary to analyze their real-world performance. We have started analyzing the performance of 5G networks in [75]; however, our evaluation of their mobile performance is still ongoing.

The choice of optimal TCP congestion control also remains a problem for cellular networks. While LTE-friendly TCP congestion control algorithms have been the topic of several papers [76, 77, 78], they have never become widely used. With BBRv2, Google has attempted to alleviate BBR's shortcomings, especially concerning its unfairness towards loss-based algorithms. An extensive analysis of BBRv2's performance in LTE and NR is still necessary once it is fully released.

Finally, more work is needed to develop dependable applications for mobile scenarios. In addition to our work, other works like [79, 80] have also looked at how the knowledge about the network can be used in making mobile applications more dependable. Still, more work is needed that focuses on the dependability and the quality of user experience.

Part II

APPENDIX

BIBLIOGRAPHY

[1] "Statista: Percentage of mobile device website traffic worldwide from 1st quarter 2015 to 2nd quarter 2022." https://www.statista.com/statistics/277125/share-of-website-traffic-coming-from-mobile-devices/.

[2] J. Ylipulli, T. Suopajärvi, T. Ojala, V. Kostakos, and H. Kukka, "Municipal WiFi and interactive displays: Appropriation of new technologies in public urban spaces," *Technological Forecasting and Social Change*, vol. 89, pp. 145–160, 2014.

[3] H. C. Jassem, "Municipal WiFi: The Coda," *Journal of Urban Technology*, vol. 17, no. 2, pp. 3–20, 2010.

[4] E. Dahlman, S. Parkvall, and J. Skold, *4G, LTE-Advanced Pro and the Road to 5G*. Academic Press, 2016.

[5] 3GPP specification TS 36.101, "Evolved Universal Terrestrial Radio Access (E-UTRA); User Equipment (UE) radio transmission and reception," Nov. 2014. Release 12, version 12.5.0.

[6] N. Becker, A. Rizk, and M. Fidler, "A measurement study on the application-level performance of LTE," in *IFIP Networking Conference*, IEEE, 2014.

[7] 3GPP specification TS 36.213, "Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures," Apr. 2018. Release 14, version 14.6.0.

[8] "srsRAN." https://docs.srsran.com/en/latest/feature_list.html.

[9] J. Strauss, D. Katabi, and F. Kaashoek, "A measurement study of available bandwidth estimation tools," in *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pp. 39–44, 2003.

[10] S. Keshav, "A Control-Theoretic Approach to Flow Control," in *Proceedings of the Conference on Communications Architecture & Protocols*, SIGCOMM '91, (New York, NY, USA), pp. 3–15, Association for Computing Machinery, 1991.

[11] V. Paxson, "End-to-End Internet Packet Dynamics," in *Proceedings of the ACM SIGCOMM '97 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '97, (New York, NY, USA), pp. 139–152, Association for Computing Machinery, 1997.

[12] C. Dovrolis, P. Ramanathan, and D. Moore, "What do packet dispersion techniques measure?," in *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213)*, vol. 2, pp. 905–914 vol.2, 2001.

[13] V. Ribeiro, R. Riedi, J. Navrátil, and L. Cottrell, "pathChirp: Efficient Available Bandwidth Estimation for Network Paths," *Proceedings of Passive and Active Measurement Workshop*, 04 2003.

[14] M. Jain and C. Dovrolis, "Pathload: A Measurement Tool for End-to-End Available Bandwidth," in *In Proceedings of Passive and Active Measurements (PAM) Workshop*, pp. 14–25, 2002.

[15] J. Strauss, D. Katabi, and F. Kaashoek, "A Measurement Study of Available Bandwidth Estimation Tools," in *Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement*, IMC '03, (New York, NY, USA), pp. 39–44, Association for Computing Machinery, 2003.

[16] B. Melander, M. Bjorkman, and P. Gunningberg, "A new end-to-end probing and analysis method for estimating bandwidth bottlenecks," in *Globecom '00 - IEEE. Global Telecommunications Conference. Conference Record (Cat. No.00CH37137)*, vol. 1, pp. 415–420 vol.1, 2000.

[17] A. Johnsson, B. Melander, M. Björkman, and M. Bjorkman, "DietTopp: A first implementation and evaluation of a simplified bandwidth measurement method," in *Second Swedish National Computer Networking Workshop*, vol. 5, Citeseer, 2004.

[18] M. Bredel and M. Fidler, "A measurement study of bandwidth estimation in IEEE 802.11 g wireless LANs using the DCF," in *International Conference on Research in Networking*, pp. 314–325, Springer, 2008.

[19] F. Michelinakis, N. Bui, G. Fioravantti, J. Widmer, F. Kaup, and D. Hausheer, "Lightweight mobile bandwidth availability measurement," in *IFIP Networking Conference (IFIP Networking)*, pp. 1–9, IEEE, 2015.

[20] M. Akselrod, N. Becker, M. Fidler, and R. Lübben, "4G LTE on the Road - What Impacts Download Speeds Most?," in *IEEE 86th Vehicular Technology Conference (VTC2017-Fall)*, 2017.

[21] "Sierra Wireless AirPrime MC7304." https://source.sierrawireless.com/devices/mc-series/mc7304/.

[22] 3GPP specification TS 36.322, "Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Link Control (RLC) protocol specification," July 2010. Release 8, version 8.8.0.

[23] "PC Engines apu3c4 system board." https://www.pcengines.ch/apu3c4.htm.

[24] "Sierra Wireless AirPrime EM7565." https://www.sierrawireless.com/iot-modules/4g-modules/em7565/.

[25] "Panorama Antennas LGMM-7-27-24-58 MIMO antenna." https://www.panorama-antennas.com/site/2G-3G-4G-5G-LTE-Antennas?product_id=70.

[26] "OpenStreetMap." https://www.openstreetmap.org/copyright/en.

[27] 3GPP specification TS 36.106, "Evolved Universal Terrestrial Radio Access (E-UTRA); FDD repeater radio transmission and reception," Jan. 2011. Release 9, version 9.3.0.

[28] "spectrummonitoring.com." http://www.spectrummonitoring.com/.

[29] 3GPP specification TS 36.331, "Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Resource Control (RRC); Protocol specification," Jan. 2011. Release 10, version 10.0.0.

[30] S. Stefania, T. Issam, and B. Matthew, "LTE, the UMTS long term evolution: from theory to practice," *Wiley*, 2009.

[31] J. Cohen, "Statistical power analysis," *Current directions in psychological science*, vol. 1, no. 3, pp. 98–101, 1992.

[32] J. Cainey, B. Gill, S. Johnston, J. Robinson, and S. Westwood, "Modelling download throughput of LTE networks," in *IEEE 39th Conference on Local Computer Networks Workshops (LCN Workshops)*, pp. 623–628, 2014.

[33] M. Akselrod and M. Fidler, "TCP Congestion Control Performance on a Highway in a Live LTE Network," in *IEEE 92nd Vehicular Technology Conference (VTC2020-Fall)*, 2020.

[34] V. Jacobson, "Congestion avoidance and control," in *ACM SIGCOMM Comput. Commun. Rev.*, vol. 18, pp. 314–329, 1988.

[35] S. Ha, I. Rhee, and L. Xu, "CUBIC: a new TCP-friendly high-speed TCP variant," *ACM SIGOPS Operating Systems Review*, vol. 42, no. 5, pp. 64–74, 2008.

[36] L. S. Brakmo and L. L. Peterson, "TCP Vegas: End to end congestion avoidance on a global Internet," *IEEE Journal on selected Areas in communications*, vol. 13, no. 8, pp. 1465–1480, 1995.

[37] K. Tan, J. Song, Q. Zhang, and M. Sridharan, "A Compound TCP Approach for High-Speed and Long Distance Networks," in *IEEE INFOCOM*, April 2006.

[38] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-Based Congestion Control," *ACM Queue*, vol. 14, pp. 20–53, Oct. 2016.

[39] P. Yang, J. Shao, W. Luo, L. Xu, J. Deogun, and Y. Lu, "TCP Congestion Avoidance Algorithm Identification," *IEEE/ACM Trans. Netw.*, vol. 22, pp. 1311–1324, Aug 2014.

[40] S. Ha and I. Rhee, "Taming the elephants: New TCP slow start," *Computer Networks*, vol. 55, no. 9, pp. 2092–2110, 2011.

[41] S. Ha and I. Rhee, "Taming the Elephants: New TCP Slow Start," *Comput. Netw.*, vol. 55, pp. 2092–2110, June 2011.

[42] E. Atxutegi, F. Liberal, K. Grinnemo, A. Brunstrom, A. Arvidsson, and R. Robert, "TCP behaviour in LTE: Impact of flow start-up and mobility," in *IFIP WMNC*, pp. 73–80, July 2016.

[43] H. Jiang, Y. Wang, K. Lee, and I. Rhee, "Tackling Bufferbloat in 3G/4G Networks," in *2012 Internet Measurement Conference*, 2012.

[44] L. Kleinrock, "Power and deterministic rules of thumb for probabilistic problems in computer communications," in *ICC'79; International Conference on Communications, Volume 3*, vol. 3, pp. 43–1, 1979.

[45] F. Li, J. W. Chung, X. Jiang, and M. Claypool, "TCP CUBIC versus BBR on the Highway," in *PAM 2018*, pp. 269–280, 2018.

[46] J. Garcia, S. Alfredsson, and A. Brunstrom, "A measurement based study of TCP protocol efficiency in cellular networks," in *WiOpt*, May 2014.

[47] R. Robert, E. Atxutegi, A. Arvidsson, F. Liberal, A. Brunstrom, and K. Grinnemo, "Behaviour of Common TCP Variants over LTE," in *IEEE GLOBECOM*, Dec 2016.

[48] K. Liu and J. Y. B. Lee, "On Improving TCP Performance over Mobile Data Networks," *IEEE Trans. Mobile Comput.*, vol. 15, pp. 2522–2536, Oct 2016.

[49] J. Wang, Y. Zheng, Y. Ni, C. Xu, F. Qian, W. Li, W. Jiang, Y. Cheng, Z. Cheng, Y. Li, *et al.*, "An Active-Passive Measurement Study of TCP Performance over LTE on High-speed Rails," in *ACM MobiCom*, 2019.

[50] L. Li, K. Xu, D. Wang, C. Peng, K. Zheng, R. Mijumbi, and Q. Xiao, "A Longitudinal Measurement Study of TCP Performance and Behavior in 3G/4G Networks Over High Speed Rails," *IEEE/ACM Trans. Netw.*, vol. 25, pp. 2195–2208, Aug 2017.

[51] S. Lee, S. Hyeon, J. Kim, H. Roh, and W. Lee, "The useful impact of carrier aggregation: A measurement study in South Korea for commercial LTE-Advanced networks," *IEEE Vehicular Technology Magazine*, vol. 12, no. 1, pp. 55–62, 2017.

[52] N. Ludant, N. Bui, A. G. Armada, and J. Widmer, "Data-driven performance evaluation of carrier aggregation in LTE-Advanced," in *IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pp. 1–6, IEEE, 2017.

[53] M. Neri, M.-G. Di Benedetto, T. Pecorella, C. Carlini, A. Castellani, P. Obino, and P. Sciarratta, "Ultra-broadband mobile networks from LTE-Advanced to 5G: evaluation of massive MIMO and multi-carrier aggregation effectiveness," in *AEIT International Annual Conference*, pp. 1–6, IEEE, 2017.

[54] M. Akselrod, "Application Level Performance of Carrier Aggregation in a Live LTE Network," in *IEEE 92nd Vehicular Technology Conference (VTC2020-Fall)*, 2020.

[55] T. Li, T. Nobukiyo, and T. Onishi, "A TCP Enhancement Method Based on LTE Carrier Aggregation Usage Identification," in *IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pp. 1–7, IEEE, 2018.

[56] M. Akselrod and M. Fidler, "Statistical delay bounds for automatic repeat request protocols with pipelining," *Performance Evaluation*, vol. 135, 2019.

[57] M. Akselrod, M. Fidler, and R. Lübben, "Stochastic Guarantees for Rate-Adaptive Streaming," in *30th International Teletraffic Congress (ITC 30)*, vol. 02, pp. 19–24, 2018.

[58] Ericsson, "Ericsson Mobility Report," 2022.

[59] "Apple's HTTP Live Streaming." https://developer.apple.com/streaming/.

[60] T. Stockhammer, "Dynamic adaptive streaming over HTTP- standards and design principles," in *Proc. of ACM MMSys*, 2011.

[61] L. De Cicco, V. Caldaralo, V. Palmisano, and S. Mascolo, "ELASTIC: a Client-side Controller for Dynamic Adaptive Streaming over HTTP (DASH)," in *Packet Video Workshop (PV)*, 2013.

[62] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A. C. Begen, and D. Oran, "Probe and adapt: Rate adaptation for HTTP video streaming at scale," *IEEE JSAC*, vol. 32, pp. 719–733, Apr. 2014.

[63] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "BOLA: Near-optimal bitrate adaptation for online videos," in *Proc. of IEEE INFOCOM*, 2016.

[64] T. C. Thang, H. T. Le, A. T. Pham, and Y. M. Ro, "An evaluation of bitrate adaptation methods for HTTP live streaming," *IEEE JSAC*, vol. 32, pp. 693–705, Apr. 2014.

[65] H. Al-Zubaidy, J. Liebeherr, and A. Burchard, "Network-Layer Performance Analysis of Multihop Fading Channels," *IEEE Trans. on Networking*, vol. 24, pp. 204–217, Feb. 2016.

[66] A. Rizk and M. Fidler, "Queue-aware uplink scheduling with stochastic guarantees," *Computer Communications*, vol. 84, pp. 63–72, Jun. 2016.

[67] H. Al-Zubaidy, V. Fodor, G. Dán, and M. Flierl, "Reliable Video Streaming With Strict Playout Deadline in Multihop Wireless Networks," *IEEE Trans. on Multimedia*, vol. 19, pp. 2238–2251, Aug. 2017.

[68] M. Fidler and J. B. Schmitt, "On the Way to a Distributed Systems Calculus: A Network Calculus with Data Scaling," in *Proc. of ACM SIGMETRICS*, pp. 287–298, June 2006.

[69] J. L. Doob, *Stochastic processes*. Wiley, 1953.

[70] J. F. C. Kingman, "A martingale inequality in the theory of queues," *Math. Proc. Cambridge*, vol. 60, pp. 359–361, Apr. 1964.

[71] Y. Jiang, "A note on applying stochastic network calculus," in *Proc. of SIGCOMM*, pp. 16–20, 2010.

[72] Y. Jiang, "Network calculus and queueing theory: Two sides of one coin," in *VALUETOOLS*, 2009.

[73] M. Fidler, B. Walker, and Y. Jiang, "Non-asymptotic delay bounds for multi-server systems with synchronization constraints," *IEEE Trans. on Parallel and Distributed Systems*, Jan. 2018.

[74] F. Ciucu and J. Schmitt, "Perspectives on network calculus: no free lunch, but still good value," in *Proc. of ACM SIGCOMM*, pp. 311–322, Aug. 2012.

[75] L. Prause and M. Akselrod, "A Measurement Study on the Application-level Performance of NSA-NR," in *IEEE 95th Vehicular Technology Conference (VTC2022-Spring)*, 2022.

[76] Y. Zaki, T. Pötsch, J. Chen, L. Subramanian, and C. Görg, "Adaptive congestion control for unpredictable cellular networks," in *Proceedings of the ACM Conference on Special Interest Group on Data Communication*, pp. 509–522, 2015.

[77] S. Abbasloo, T. Li, Y. Xu, and H. J. Chao, "Cellular controlled delay TCP (C2TCP)," in *IFIP Networking Conference (IFIP Networking) and Workshops*, pp. 118–126, IEEE, 2018.

[78] S. Abbasloo, C.-Y. Yen, and H. J. Chao, "Wanna make your TCP scheme great for cellular networks? Let machines do it for you!," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 1, pp. 265–279, 2020.

[79] J. W. Kleinrouweler, B. Meixner, J. Bosman, H. Van Den Berg, R. Van Der Mei, and P. Cesar, "Improving mobile video quality through predictive channel quality based buffering," in *30th International Teletraffic Congress (ITC 30)*, vol. 1, pp. 236–244, IEEE, 2018.

[80] C. Moldovan, F. Wamser, and T. Hoßfeld, "Energy-Efficient Adaptation Logic for HTTP Streaming in Mobile Networks," in *International Conference on Networked Systems (NetSys)*, IEEE, 2019.

PUBLICATIONS

[1] L. Prause and M. Akselrod, "A Measurement Study on the Application-level Performance of NSA-NR," in *IEEE 95th Vehicular Technology Conference (VTC2022-Spring)*, 2022.

[2] V. A. Vu and M. Akselrod, "An experiment of dual-LTE MPTCP with In-Car Voice Assistant," in *IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, 2021.

[3] M. Akselrod, "Application Level Performance of Carrier Aggregation in a Live LTE Network," in *IEEE 92nd Vehicular Technology Conference (VTC2020-Fall)*, 2020.

[4] M. Akselrod and M. Fidler, "TCP Congestion Control Performance on a Highway in a Live LTE Network," in *IEEE 92nd Vehicular Technology Conference (VTC2020-Fall)*, 2020.

[5] M. Akselrod and M. Fidler, "Statistical delay bounds for automatic repeat request protocols with pipelining," *Performance Evaluation*, vol. 135, 2019.

[6] M. Akselrod, M. Fidler, and R. Lübben, "Stochastic Guarantees for Rate-Adaptive Streaming," in *30th International Teletraffic Congress (ITC 30)*, vol. 02, pp. 19–24, 2018.

[7] M. Akselrod and M. Fidler, "Statistical Delay Bounds for Automatic Repeat Request Protocols with Pipelining," in *30th International Teletraffic Congress (ITC 30)*, vol. 01, pp. 10–18, 2018.

[8] M. Akselrod, N. Becker, M. Fidler, and R. Lübben, "4G LTE on the Road - What Impacts Download Speeds Most?," in *IEEE 86th Vehicular Technology Conference (VTC2017-Fall)*, 2017.

SCIENTIFIC CAREER

Contact mark.akselrod@ikt.uni-hannover.de

Work Experience

04/2016 - 03/2023 **Research Assistant**, Leibniz Universität Hannover, Institute of Communications Technology (IKT)

Research in areas of mobile and vehicular communication

03/2015 - 03/2016 **Software Developer**, jService GmbH

Software development for VW infotainment systems

04/2013 - 03/2014 **Student Assistant**, Leibniz Universität Hannover, Institute of Communications Technology (IKT)

Software development for the student lab *Networks and Protocols*

Education

10/2012 - 11/2014 **M. Sc. Computer Science**, Leibniz Universität Hannover

10/2009 - 09/2012 **B. Sc. Computer Science**, Leibniz Universität Hannover

Awards

2018 **Best Paper Award**

Mark Akselrod and Markus Fidler: "Statistical Delay Bounds for Automatic Repeat Request Protocols with Pipelining", 30th International Teletraffic Congress

10/2011 - 9/2014 **Deutschlandstipendium**

2010 **Niedersachsenstipendium**