uc3m | Universidad Carlos III de Madrid

TESIS DOCTORAL

Optimizing the Delivery of Multimedia over Mobile Networks

Autor: Foivos Ioannis Michelinakis, IMDEA Networks Institute
University Carlos III de Madrid
Director \Tutor: Joerg Widmer, IMDEA Networks Institute

DEPARTAMENTO DE INGENIERÍA TELEMÁTICA

Leganés (Madrid), Junio de 2018

uc3m | Universidad Carlos III de Madrid

PH.D. THESIS

Optimizing the Delivery of Multimedia over Mobile Networks

Autor: Foivos Ioannis Michelinakis, IMDEA Networks Institute
University Carlos III de Madrid
Director \Tutor: Joerg Widmer, IMDEA Networks Institute

DEPARTMENT OF TELEMATIC ENGINEERING

Leganés (Madrid), June 2018

Optimizing the Delivery of Multimedia over Mobile Networks

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Prepared by
Foivos Ioannis Michelinakis, IMDEA Networks Institute, University Carlos III of Madrid

Under the advice of
Joerg Widmer, IMDEA Networks Institute
Departamento de Ingeniería Telemática, Universidad Carlos III de Madrid

---

Date: Junio, 2018

Web/contact: foivos.michelinakis@imdea.org

This work has been supported by IMDEA Networks Institute.

TESIS DOCTORAL

OPTIMIZING THE DELIVERY OF MULTIMEDIA OVER MOBILE NETWORKS

Autor: Foivos Ioannis Michelinakis, IMDEA Networks Institute
University Carlos III de Madrid
Director \Tutor: Joerg Widmer, IMDEA Networks Institute

Firma del tribunal calificador:

Presidente:

Vocal:

Secretario:

Calificación:

Leganés, de de

Disclaimer

We would like to emphasize that this is a presentation of academic research and should not be taken as indicative of Spotify's product plans.

The European Commission is not responsible for any use that may be made of this research.

Part of this work was carried out while Foivos Michelinakis was visiting:

- Korea Advanced Institute of Science and Technology (KAIST)
- Simula Research Laboratory (Simula)
- Spotify AB (Spotify)

Acknowledgements

The front page of this PhD thesis may have just a single name as author but a lot of people have contributed in its creation. Their contributions range from technical through ethical to just "being there". For example, my family was always by my side, even though physically they were far away.

I am grateful to my supervisor Dr. Joerg Widmer who gave me the opportunity pursue a doctorate in one of the fastest growing and interesting engineering areas: mobile communications. His guidance and mostly patience have been invaluable in developing my profile as an engineer and helping me grow as a person. He was not alone in this effort though. I would also like to express my gratitude to my supervisors in the three internships I had during my PhD studies: Gunnar Kreitz in Spotify, Yung Yi in KAIST and Ozgu Alay in Simula. Each of them had their unique contribution to my growth and gave me the opportunity of being exposed to a variety of research and work environments. From a "work hard, party hard" hip startup, where I was constantly surrounded by extremely gifted people making me often think I am the stupidest person in the room, to witnessing first hand the Korean work ethics, working with a very dedicated team and to the family friendly, casual, but very efficient Scandinavian workplace.

During this PhD trip I had the pleasure to work closely with a number of wonderful people. Nicola, Fabian, Andra and Hossein have been excellent co-authors and friends. Without their contributions none of my papers would have been published. Guido has been the best intern I could be paired with and was always a joy to supervise him. Miguel has been an excellent and incredibly patient engineer.

A special thank you is deserved to my colleagues at IMDEA Networks Institute for welcoming me to a foreign country and being the perfect workmates and friends. Some of them are Omar, Miriam and Jorge who helped me adapt in Spain, even going with me to visit houses and being my go-to translators. Angelos, Evgenia, Elli, Nicholas, Vasilis and Stelios who helped me keep my Greek fresh and even gave me the opportunity to become godfather to a wonderful girl. Javier and Rosa who went well beyond their duties to help me with any difficulties I had living in Spain. Javier was even accompanying me to the hospital the day of my knee surgery. Roderick has been a wonderful and calm colleague, friend and flatmate. Christian, Aymen, Mohamed, Maurizio, Amr, Hany, Dario, Guillermo, Roberto, Alejandro, Lisa, Ander, Julien, Gines, Pablo, Joan, Noelia, Syed, Camilo, Allyson, Arash, Thomas, Claudio, Adrian, Danilo..... and so many more have

been mostly friends than coworkers. I am sure I am forgetting someone, but you know you are important to me.

Finally, I have to thank the rest of the people I met and worked with during my PhD trip at my internships: Riccardo Petrocco, Boxun Zhang and the metadata quality squad who hosted me in their space at the Spotify headquarters, Boram Jin and the rest of the Lanada group at KAIST and Giorgos, Kostas, Cise and Isabel at the mobile systems and analytics group at Simula.

My only regret and advice to the new people is how late I discovered the productivity gains that a good pair of noise canceling headphones paired with a brown noise generator can offer in a lively open space environment such as IMDEA.

Abstract

The consumption of multimedia content is moving from a residential environment to mobile phones. Mobile data traffic, driven mostly by video demand, is increasing rapidly and wireless spectrum is becoming a more and more scarce resource. This makes it highly important to operate mobile networks efficiently. To tackle this, recent developments in anticipatory networking schemes make it possible to to predict the future capacity of mobile devices and optimize the allocation of the limited wireless resources. Further, optimizing Quality of Experience—smooth, quick, and high quality playback—is more difficult in the mobile setting, due to the highly dynamic nature of wireless links. A key requirement for achieving, both anticipatory networking schemes and QoE optimization, is estimating the available bandwidth of mobile devices. Ideally, this should be done quickly and with low overhead.

In summary, we propose a series of improvements to the delivery of multimedia over mobile networks. We do so, be identifying inefficiencies in the interconnection of mobile operators with the servers hosting content, propose an algorithm to opportunistically create frequent capacity estimations suitable for use in resource optimization solutions and finally propose another algorithm able to estimate the bandwidth class of a device based on minimal traffic in order to identify the ideal streaming quality its connection may support before commencing playback.

The main body of this thesis proposes two lightweight algorithms designed to provide bandwidth estimations under the high constraints of the mobile environment, such as and most notably the usually very limited traffic quota. To do so, we begin with providing a thorough overview of the communication path between a content server and a mobile device. We continue with analysing how accurate smartphone measurements can be and also go in depth identifying the various artifacts adding noise to the fidelity of on device measurements. Then, we first propose a novel lightweight measurement technique that can be used as a basis for advanced resource optimization algorithms to be run on mobile phones. Our main idea leverages an original packet dispersion based technique to estimate per user capacity. This allows passive measurements by just sampling the existing mobile traffic. Our technique is able to efficiently filter outliers introduced by mobile network schedulers and phone hardware. In order to asses and verify our measurement technique, we apply it to a diverse dataset generated by both extensive simulations and a week-long measurement campaign spanning two cities in two countries, different radio technologies, and covering all times of the day. The results demonstrate that our technique is

effective even if it is provided only with a small fraction of the exchanged packets of a flow. The only requirement for the input data is that it should consist of a few consecutive packets that are gathered periodically. This makes the measurement algorithm a good candidate for inclusion in OS libraries to allow for advanced resource optimization and application-level traffic scheduling, based on current and predicted future user capacity.

We proceed with another algorithm that takes advantage of the traffic generated by short-lived TCP connections, which form the majority of the mobile connections, to passively estimate the currently available bandwidth class. Our algorithm is able to extract useful information even if the TCP connection never exits the slow start phase. To the best of our knowledge, no other solution can operate with such constrained input. Our estimation method is able to achieve good precision despite artifacts introduced by the slow start behavior of TCP, mobile scheduler and phone hardware. We evaluate our solution against traces collected in 4 European countries. Furthermore, the small footprint of our algorithm allows its deployment on resource limited devices.

Finally, in an attempt to face the rapid traffic increase, mobile application developers outsource their cloud infrastructure deployment and content delivery to cloud computing services and content delivery networks. Studying how these services, which we collectively denote Cloud Service Providers (CSPs), perform over Mobile Network Operators (MNOs) is crucial to understanding some of the performance limitations of today's mobile apps. To that end, we perform the first empirical study of the complex dynamics between applications, MNOs and CSPs. First, we use real mobile app traffic traces that we gathered through a global crowdsourcing campaign to identify the most prevalent CSPs supporting today's mobile Internet. Then, we investigate how well these services interconnect with major European MNOs at a topological level, and measure their performance over European MNO networks through a month-long measurement campaign on the MONROE mobile broadband testbed. We discover that the top 6 most prevalent CSPs are used by 85% of apps, and observe significant differences in their performance across different MNOs due to the nature of their services, peering relationships with MNOs, and deployment strategies. We also find that CSP performance in MNOs is affected by inflated path length, roaming, and presence of middleboxes, but not influenced by the choice of DNS resolver. We also observe that the choice of operator's Point of Presence (PoP) may inflate by at least 20% the delay towards popular websites.

Table of Contents

List of Tables

List of Figures

List of Acronyms

C-RNTI Cell Radio Network Temporary Identifier

DRX Discontinuous Reception

EPC Evolved Packet Core

HARQ Hybrid Automatic Repeat Request

IMSI International Mobile Subscriber Identity

MCS Modulation and Coding Scheme

PDCP Packet Data Convergence Protocol

QoE Quality of Experience

QoS Quality of Service

RLC Radio Link Control

RRC Radio Resource Control

TB Transport Block

TFT Traffic Flow Template

UE User Equipment

DMA Direct Memory Access

UID User Identifier

SoC System on Chip

SKB Socket Buffer Structure

NAPI New API

NIC Network Interface Controller / Network Interface Card

WLAN Wireless Local Area Network

SKBs Socket Buffer Structures

PGW Packet Data network Gateway

NR New radio

SGW Serving Gateway

TTI Transmission Time Interval

FDD Frequency-Division Duplexing

TDD Time-Division Duplexing

CDNs Content Distribution Networks

IXP Internet Exchange Point

PoP Point of Presence

RBs Resource Blocks

MAC Medium Access Control

RBG Resource Block Group

SIM Subscriber Identification Module

OWL Online Watcher for LTE

RAN Radio Access Network

CBR Constant Bitrate

CSPs Cloud Service Providers

CSP Cloud Service Provider

MNOs Mobile Network Operators

EEA European Economic Area

EU European Union

TTFB Time to First Byte

RTT Round Trip Time

PEP Performance Enhancing Proxies

NAT Network address translation

EPDF Empirical Probability Density Function

Part I

Introduction

Chapter 1

Introduction

Modern western civilization relies on "information". Information is needed to understand the world around us, communicate and make informed decisions. Smartphones are an example of how it enhances modern life. Smartphones are hand-held computers focusing primarily on various forms of communication. They enable users to among others, access text, audio, images and video, hosted on other computers, called servers, through a network connection. These forms of content are collectively called "multimedia". Multimedia files are broken into smaller pieces by the server and transmitted in sequence to the smartphone. A sequence of packets forms a (network) flow[1]. In this thesis, we will explore ways to optimize the way smartphones make use of "information", which in our context are multimedia flows. As a fortunate side effect, the same tools and techniques might have a favorable influence on the "energy" aspect of these devices, which is battery life. In summary, we will study the properties of these flows, how they are affected in the various stages of transmission and propose ways to exploit the information we can gather from them to make them more efficient. Since we cannot study flows in isolation, we will also study the various network components that interact with them and how these components are interconnected. Thus, we will study the smartphone itself, the radio access network, core network components and the servers hosting the multimedia.

1.1. Motivation

Over the last few years, emerging new technologies and devices have altered the traditional scheme of media discovery and delivery. The gigantic growth in popularity of online Social Networks (OSNs), Content Distribution Networks (CDNs), video sharing websites, hand-held devices with multimedia capabilities and access to fast 4G/3G networks have shape-shifted the old media dissemination model.

The availability of affordable mobile broadband connections encourages people to consume a lot of multimedia content, often HD videos, on their handheld devices. The global mobile data

[1] A stricter definition of "flow" will be given later.

traffic is expected to increase sevenfold between 2016 and 2021, reaching 49.0 exabytes per month by 2021 [3], mostly caused by the ever increasing video traffic. Augmenting the network capacity to cope with this traffic or offloading the traffic to other wireless technologies such as WLAN, may not always be the best solution since the former is very costly and the latter may not work in all cases [4, 5]. Even though spectrum efficiency is improving thanks to the fifth generation [6] of mobile networks, the wireless medium is becoming a scarcer and scarcer resource, due to the ever increasing demand for mobile communication. As a consequence, it is highly important to try to reduce traffic volume and increase network efficiency. Recently, a number of papers addressed improved resource allocation mechanisms based on capacity prediction techniques. For instance, [7–10] propose to use resources when they are more abundant and cheap, and to refrain from or to limit communication when it is more expensive (e.g., lower spectral efficiency, higher congestion, etc.) by exploiting perfect knowledge of the future capacity.

In Appendix A, we will survey the state of the art on mobile capacity prediction techniques and build a model for both short and medium to long term prediction errors in order to be able to quantify the impact of prediction uncertainties in resource allocation. Most short term prediction techniques [11, 12] rely on time series filtering solutions, such as moving average and autoregressive (ARMA) or autoregressive conditional heteroskedasticity (ARCH) modeling. Thus, in order to allocate resources on a given time granularity, prediction must be available with the same granularity and, consequently, mobiles must be able to measure capacity with the same granularity [13]. In other words, they all share the requirement of having as many as possible past values of bandwidth.

Apart from mobile operators content providers are affected as well. The distribution of audio and video content is swiftly moving from the desktop environment to the mobile one. This shift affects streaming applications like Spotify [14], that have seen an increasing mobile consumption over the last years and continue to do so. In the competitive market of streaming services, Quality of Experience (QoE) is important. The metrics comprising QoE include fast start-up time, low playback latency, stutter-free media delivery, and high bitrates.

Given the high variation of the available radio resources and capabilities of mobile devices, streams are often available in different bitrates. To achieve high QoE, it is imperative to select the highest possible bitrate, as constrained by available bandwidth and device hardware. Ideally, it is desirable to avoid switching bitrates during playback, but due to the difficulty of predicting bandwidth, it is currently a common practice to begin playback on a low-quality bitrate, and then increase it if possible. In order to improve on the current practice and select an appropriate bitrate also for the initial part of stream, it is required to have a bandwidth estimation algorithm that operates on very small amounts of traffic.

Another key use case for a rapid bandwidth estimator is to make buffering decisions. Streaming media players need to buffer content before commencing playback, and thus need to decide on the size of said buffer. Such decisions require an estimate of the bandwidth available in the near future. This is particularly important for applications such as Spotify, which do not perform bi-

trate switching, but still require low playback latency. Furthermore, when streaming audio tracks, the small amount of data transferred makes it difficult to apply traditional bandwidth estimators. Spotify reported in a previous study [14] a median playback latency of 265 ms with less than 1% of streams suffering one or more stutter events. However, at that point, most users were streaming on desktops rather than mobile devices, and the data reported did not include phones.

Part II of this thesis is dedicated to estimating the bandwidth of mobile users in minimally intrusive ways. As we will analyse in later chapters, getting a good enough estimation of mobile bandwidth is very challenging. Especially, if the estimations should be generated with the frequency required by traffic prediction algorithms. Some of the limitations that we have to take into account include:

- **Low CPU cost.** Even modern mobile devices have limited CPU resources, if we take into account user demands (*i.e.*, high density displays, multitasking etc.). Our algorithms have very low complexity, thus do not cost a lot of CPU cycles.

- **No traffic generation.** The most popular approach for estimating bandwidth is generating saturation traffic and observing how much traffic actually "goes through". Since most users have a fixed quota of traffic per month, we had to invent solutions that follow a very different logic in order to avoid generating traffic. We focus on inferring bandwidth by cleverly analysing the arrival pattern of the traffic other applications are generating.

- **Minimal impact on battery.** Battery is the most scarce resource of a smart phone. Our algorithms take this into account. Radio is the most resource intensive component of a smartphone and since our algorithms do not generate traffic at all, do not contribute to having it active. Also, the low complexity of our solutions has the side effect of not requiring a lot of energy to run.

- **Fast execution.** It is critical to generate an estimate fast, so that the traffic prediction algorithms have "up to date" data to work with. Our algorithms are able to give a result a few ms after a sample has been received and timestamped by the kernel of the device.

In the following chapters we propose two mobile bandwidth estimation solutions that respect the above limitations.

Naturally, the question regarding how trustworthy are measurements performed in a device as constrained as the mobile phone arises. There is extensive past work in attempting to measure the characteristics, such as speed, of mobile connections [15–19]. Despite that, and very surprisingly, at the time of conducting the research presented in Part II, there was no study evaluating the accuracy of smartphone measurements over mobile networks. At the same time, we also noticed that different phones may exhibit very different behavior in aspects that are easily observable, such as end-to-end throughput, under the same conditions.

To address this, in Chapter 4, we provide a detailed study of the reliability of mobile phone measurements and further analyse how different chipsets impact on their accuracy. We use an

LTE control channel decoder called Online Watcher for LTE (OWL) [20], to compare the bandwidth observed at the kernel level to the actual bandwidth measured at the physical layer. OWL has excellent reliability (*i.e.*, successful decoding rate higher than 99.85%) which is fundamental to provide ground truth readings of LTE scheduling information. OWL decodes LTE control messages sent by base stations to mobile devices[2]. Therefore, we can observe traffic at the physical layer of the device. Then, we compare the view we have of the physical layer, with the view we obtain from higher layers of the device like the kernel and the application. Further, Chapter 5 analyses in detail the measurement artifacts that add noise in smartphone measurements performed using tcpdump, such as the techniques proposed in Chapters 6 and 7.

Apart from the low level aspects of mobile measurements, in Part III, we also dive into how management, configuration and political decisions affect the performance of mobile networks. Mobile app developers have a wealth of tools and techniques at their disposal that help them decrease the amount of time and effort required to develop, deploy, and maintain their apps. One particularly powerful and very common technique is to use a variety of third-party online services such as on-demand cloud computing platforms (*e.g.*, Amazon Web Services) and content delivery networks (*e.g.*, Akamai) in their apps. This technique makes it easier and more efficient to deploy mobile apps at a global scale by shifting the burden of managing and maintaining server infrastructure from app developers to these Cloud Service Providers (CSPs).

While there have been quite a few studies on the implications of newer protocols such as QUIC on the performance of mobile apps [21], there is a notable lack of a systematic study on the performance of third-party cloud computing and content delivery services used by mobile apps. In fact, the relationships between CSPs, app developers, and Mobile Network Operators (MNOs) are tangled, and decisions made by each entity can have a significant and far-reaching impact on the ecosystem as a whole. For example, MNOs may peer with popular CSPs to reduce their costs and improve performance for their users, thereby placing these CSPs and the apps that use them at an advantage over others.

Chapter 8 characterizes the performance of these services in the wild. Such evaluation is critical in understanding, and ultimately reducing, the technological gap between the limited capabilities of the mobile Internet infrastructure and the performance requirements of current and future mobile apps [22].

Since 15 June 2017, the European Union (EU) has ended roaming surcharges within the states participating in European Economic Area (EEA). Travelers between these countries only pay domestic charges for their mobile data. However, to the best of our knowledge there is no systematic study about the effect of roaming on the Quality of Service (QoS) of mobile applications. Chapter 8 further explores this aspect that is expected to affect millions of European citizens.

[2]Note that OWL does not violate users' privacy, since it only logs UEs' temporary identifiers.

1.2. Contributions and Published Material

The research work presented in this thesis has been published and presented in several peer-reviewed journals and conferences. In detail, the related publications are six conference papers (in three of these I am the first author and in the other three the second author) [1, 23–27] and two journal articles (one as first author and one as second author) [2, 28]. As of writing these lines another journal paper (second author) is under submission, which extends the work presented in [1]. Parts of this work have been presented as two demos [29, 30], one invited talk [31] as well as several posters. Finally, during my PhD I contributed to one peer-reviewed conference paper [32], which is out of the scope of the current thesis and thus not discussed.

Research is a collaborative process and all of these works have been the result of the joint efforts of their authors. For completeness and in order to have everything presented in its proper context, in the following chapters, I present the work in a similar fashion as it was originally published, without highlighting or isolating my own contributions. The goal of this section is to present a summary of the above publications and clarify what was my contribution to each one. In order to comply with plagiarism rules, content from papers where I am not the first author is either presented as an appendix or shortened, paraphrased and the figures' captions cite the original paper.

The main topic of this thesis is, as its title declares, "optimizing the delivery of multimedia over mobile networks". In an attempt to make it a stand alone and cohesive work, only the most relevant parts of the published papers to its main topic are presented. Some work that I did during my PhD that is not discussed in this text includes measuring the performance accuracy over WiFi and speed and delay measurements of mobile networks.

I have studied in depth the arrival patterns of IP packets to mobile devices and analysed the possible artifacts that may distort their accurate observation. To do so, I gathered an extensive set of traces in a variety of conditions and countries, as well as generated vast simulation data. Then, I proposed two algorithms that may use arrival pattern information to passively estimate bandwidth, either by analysing the first few packets of a TCP flow, or by sampling a minimal portion of the traffic generated by bigger flows. Chapter 5 presents the measurement artifacts and Chapters 6 and 7 the two algorithms. Further, Chapter 2 is mostly original content created specifically for this thesis and acts as supplementary material to the peer reviewed work presented in the other three chapters. To the best of my knowledge, there is no other public resource that analyses the whole process of delivering a packet to a mobile phone as Chapter 2, does. The available resources are scattered, outdated or wrong and some parts of the process, like the baseband operation, have notoriously limited public documentation. Thus, it is a good starting point for anyone interested in learning about mobile communications. The related publications are:

- **Foivos Michelinakis**, Nicola Bui, Guido Fioravantti, Joerg Widmer, Fabian Kaup, David Hausheer, "Lightweight Capacity Measurements For Mobile Networks", in Computer Communications, 84. pp. 73-83, June 2016, ISSN 0140-3664.

- **Foivos Michelinakis**, Gunnar Kreitz, Riccardo Petrocco, Boxun Zhang, Joerg Widmer, "Passive Mobile Bandwidth Classification Using Short Lived TCP Connections" in The 8th IFIP Wireless and Mobile Networking Conference (WMNC 2015), 5-7 October 2015, Munich, Germany.

- **Foivos Michelinakis**, Nicola Bui, Guido Fioravantti, Joerg Widmer, Fabian Kaup, David Hausheer, "Lightweight Mobile Bandwidth Availability Measurement", in The 14th IFIP Networking 2015 Conference, 20-22 May 2015, Toulouse, France.

A crucial part of the above was validating that smartphones are capable of reporting accurately enough the timing of packet arrivals. Chapter 4 analyses how we assessed smartphone measurement accuracy. In this work, I created the experiment logic and performed the experiments, as well as analyse the gathered data. I also performed experiments regarding the accuracy of measurements over the WiFi interface, which are not discussed in this thesis. The related publication, an extension of which is under submission to the "Pervasive and Mobile Computing" Journal, is:

- Nicola Bui, **Foivos Michelinakis**, Joerg Widmer, "Fine-grained LTE Radio Link Estimation for Mobile Phones", in The 18th International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM 2017), 12-15 June 2017, Macau, China.

In collaboration with researchers from TU Darmstadt I contributed to two works evaluating the QoS of mobile networks based on crowd sourced data. I worked on gathering traces from Spanish operators and analysing the dataset. Parts of this work are presented in Section 8.6. The related publications are:

- Fabian Kaup, **Foivos Michelinakis**, Nicola Bui, Joerg Widmer, Katarzyna Wac, David Hausheer, "Assessing the Implications of Cellular Network Performance on Mobile Content Access", in IEEE Transactions on Network and Service Management, 13 (2). pp. 168-180, March 2016, ISSN 1932-4537.

- Fabian Kaup, **Foivos Michelinakis**, Nicola Bui, Joerg Widmer, Katarzyna Wac, David Hausheer, "Behind the NAT – A Measurement Based Evaluation of Cellular Service Quality", in The 11th International Conference on Network and Service Management (CNSM 2015), 9–13 November 2015, Barcelona, Spain.

I performed extensive measurements and analysis of the performance of third-party services over twelve dominant European network operators, using the MONROE experiment testbed. This work is discussed in Chapter 8 and the related publication is:

- **Foivos Michelinakis**, Hossein Doroud, Abbas Razaghpanah, Andra Lutu, Narseo Vallina-Rodriguez, Phillipa Gill, Joerg Widmer, "The Cloud that Runs the Mobile Internet: A Measurement Study of Mobile Cloud Services", in The 37th IEEE International Conference on Computer Communications (IEEE INFOCOM 2018), 15-19 April 2018, Honolulu, HI, USA.

Finally, I created simulations helping define and validate a stochastic model for user throughput prediction in mobile networks. We did not discuss these simulations in the resulting publication though. Appendix A presents the model and the related publication is:

- Nicola Bui, **Foivos Michelinakis**, and Joerg Widmer, "A Model for Throughput Prediction for Mobile Users", in European Wireless May 2014, Barcelona, Spain.

Chapter 3 offers an overview of the state of the art as it was presented in the above publications.

1.3. Thesis Overview

The thesis is divided into three parts. Part I contains all the background material required to follow the flow of the thesis. Part II discusses how trustworthy are smartphone measurements and then proposes 2 bandwidth estimation algorithms. Part III presents a series of measurements which evaluate the interconnection of Cloud Service Providers and mobile operators. In particular, Part I starts with the present introduction. Then, Chapter 2 offers a very extended background overview, meant to make the information presented in the chapters about mobile bandwidth estimation more approachable to the average reader. Chapter 3 summarizes the state of the art. Part II starts with Chapter 4 which studies the accuracy of smartphone measurements. Chapter 5 provides an overview of the artifacts that distort the proper timing of packet arrivals and should be accounted for by any algorithm that relies on packet interarrival times. In Chapter 6, we propose an estimation of available bandwidth of mobile phones through passive traffic monitoring of the traffic generated during the initial few hundred milliseconds of the TCP "slow start" phase. Chapter 7 provides a simple tool that evaluates passively and with minimum impact the per user capacity variations over time in a mobile environment. Chapter 8 is the sole chapter of Part III and discusses a series of measurement studies aimed at identifying dominant Cloud Service Providers, analysing how they interconnect with mobile operators and how the interconnections and operators' configurations affect their performance. Chapter 9, concludes the thesis offering the bigger picture of the presented work and discusses some open issues. Finally, Appendix A presents a model that may take as input the bandwidth estimations of the algorithms of Chapters 6 and 7 and derives throughput predictions.

Chapter 2

Background

A big portion of the thesis analyses the arrival patterns of IP packets as they are reported by the Linux kernel. We use these patterns to infer bitrate at the physical layer, in other words, how fast a base station is transmitting data to the target smartphone. On other parts, we study how the behavior of the different players in the path between a client and a server affects the delay of the connection. This chapter serves as a brief introduction on how an IP packet travels between a server and a mobile device and how it interacts with the entities that handle it along the way. We follow the path from the moment the IP packet is transmitted from the Network Interface Controller / Network Interface Card (NIC) of the server, until its payload is delivered to the application of the smartphone requesting the related data. We put more emphasis on the steps that can have an effect on what we discuss later on. Figure 2.1 presents the main nodes along this path. To the best of our knowledge, this chapter is the only cohesive, accurate and up–to–date public resource that analyses all the parts of a packet trip between a server and a mobile device. Considering how fragmented and often surprisingly hard it is to find reliable information for some of the material discussed here, this chapter serves as an excellent starting point for anyone interested in mobile networks.

2.1. "Flow" Definition

Since we are interested in the end to end connection between two machines, we have to start by specifying what is a "flow". The term "flow" may have multiple definitions in networking. We are interested the total amount of traffic that a smartphone can receive from a specific server. Therefore, in this thesis a "flow" is a set of packets generated by a server traveling towards a client. Unless otherwise specified, we will be referring to such cases when using this term. This is in contrast to a more popular but strict definition of "flow", where it is defined as a set of packets which have fixed values in specific parts of their network and transport layer headers.

In this stricter case, a TCP flow is identified by the values of 1) protocol, 2) source IP address and 3) destination IP address of the IP header and 4) source port and 5) destination port of the

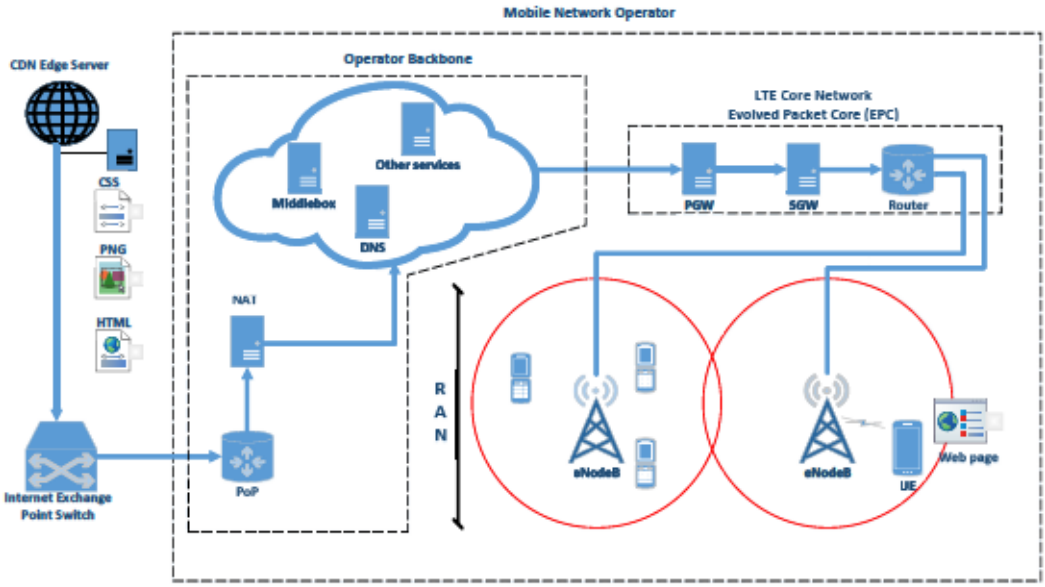


Figure 2.1: Simplified topology and network components that form the path between a CDN edge server and a mobile device.

TCP header. The same fields can be used to identify a UDP flow. For a flow generated by the `ping` command, the same fields from the IP header and the identifier field of the ICMP header define a flow. In recent years, it is common to open multiple TCP connections to a server when fetching content, such as web pages, in part to overcome the limitations of the slow start phase of TCP. In our studies, we do not distinguish packets of different TCP flows, rather all the packets of such TCP flows are treated as one big flow.

Another key aspect of "flow" definition is the vantage point where traffic is observed. In our case, it is usually at the kernel of an Android smartphone. So, the view we have of the traffic is limited by the capabilities of the device. This poses a great limitation in accurately evaluating traffic characteristics. Most specifically, it was not possible for the contemporary smartphone hardware we used in the experiments we will present in the next chapters, to accurately report the precise arrival time of IP packets. It seems that more modern hardware, as well as recent advancements in the linux kernel have the potential to increase the precision of traffic monitoring, improving the accuracy of our tools. There is no guarantee though that even with improved equipment, monitoring could be consistently precise enough to get the real view of the traffic. Ideally, we would like to have sub-millisecond accuracy for the majority of the packets at the IP layer. A big part of this thesis is dedicated to techniques that lessen the effect of such inaccuracies.

2.2. Content Servers and the Modern (Mobile) ISP

Due to the peculiarities of the HTTP and TCP protocols, the main factor dictating the performance of modern web applications and web sites is latency rather than bandwidth. The main

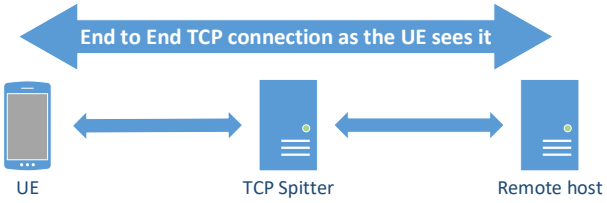


Figure 2.2: Logical representation of a TCP splitting middlebox.

contributing factor to latency is the distance between the content and the user. Distance in regards to Internet is measured in two ways. The first way is to measure the physical distance, which based on the speed of light, dictates the minimum delay of the transmission. The second way is to measure the number of nodes that have to process and forward a packet in its trip towards the user. To this end, current trends are pushing towards placing content as close to the user as possible and have led to the very high popularity of Content Distribution Networks (CDNs). CDNs are a network of proxy servers geographically distributed in order to be as close as possible to the end user.

As we will see in Chapter 8, CDNs are widely used in the mobile ecosystem. Thus, a good starting point for examining the trip of a packet between a server and a mobile client is a CDN edge server. Edge servers are the servers of a CDN closer to the user. Depending on the CDN's size and peering agreements with ISPs, edge servers can be at various locations. Typically they are located or, at least, well connected to an Internet Exchange Point (IXP), a location where ISPs and CDNs exchange traffic. If the user's ISP and the CDN are colocated at an IXP, the traffic enters the ISP's network at this point. Otherwise, the CDN uses the IXP or a transit link to send the traffic to a tier-1 ISP, which routes it to the user's ISP.

Some of the biggest CDNs put their servers inside the network of ISPs, a strategy that minimizes latency and transit costs. The same approach is followed by some ISPs which offer CDN services. In such cases, the server usually has a private IP address, since it is behind the Network address translation (NAT) device of the operator, making it hard for third parties to identify who is the true owner of the server. In the future it is expected that content and computing resources will continue to move closer to the user in a concept called "fog computing" or "mobile edge computing". In some cases, it is expected that even base stations will have storage and computing capabilities, offering the smallest possible distance between content and user.

When the packet enters the operator's network, it has to pass by a few more entities before it reaches the mobile core. The entry point to the operator is called Point of Presence (PoP), which, in case of IPv4 networks, may also be collocated with the NAT. The norm for mobile operators is to use NAT in order to be able to provide all of their customers with IPv4 addresses. This results in a big number of customers having the same public IP address. The network distinguishes them based on their private address, which is unique within the private network behind the NAT. In our experiments, we found that only a handful of operators, located in Norway and Sweden, assign

public IPs to their mobile clients. After the PoP, the operator may apply traffic shaping and / or pass the traffic through middleboxes.

A common type of middlebox in mobile networks are TCP splitters. When a user initiates a TCP connection the splitter intercepts it, replies to the user posing as the target server (*i.e.*, the packets have the IP address of the target server) and then opens a connection with the target server itself. The splitter then relays the packets between the target server and the user, as is shown in Figure 2.2. This may increase the quality of the connection since it isolates the loss and delay prone mobile part of the path (between the user and the middlebox) from the long distance part (between the middlebox and the target server) and thus allows the TCP congestion window to reach values close to the bandwidth delay product. Otherwise, losses occurring at the mobile part, would be interpreted as congestion by TCP and thus keep the congestion window low.

2.3. LTE Architecture

In this section we provide a brief overview of the main components and characteristics of mobile networks that have an effect on what we will discuss in the following chapters. We will present the architecture of the LTE communication standard, since it is the one most frequently used in this thesis and has in general widespread adoption.

2.3.1. LTE Core Network

The LTE core, widely known as Evolved Packet Core (EPC), shares a lot of similarities with the cores of other popular mobile communication standards in use today, such as the 3G family. Thus, the following is representative of their core as well. Please note though, that the following may not apply to the upcoming 5G-New radio (NR) standard. As of writing this thesis, only part of the early specification of 5G has been completed [33]. Its core seems to be focusing on offering a modularized set of services, called "network functions", making it hard to specify specific core components.

In brief, the main components of the core network of LTE are:

- **Packet Data network Gateway (PGW):** It is the connection point between the Internet and the network of the operator.

- **Serving Gateway (SGW):** It is the regional entity of LTE that forwards data packets to the eNodeBs where the destination UEs are. It does so by keeping track of the mobility of the UEs that are associated to the eNodeBs that it is connected to. Also, it is the anchor point of the UEs who perform handover between two eNodeBs. It is worth noting that the majority of the handovers take place between eNodeBs that belong to the same SGW.

- **eNodeB:** It is the base station of the LTE network. The SGW forwards there the packets that are destined to the UEs that are connected to it. Upon reception of a packet, it

is temporarily stored and a scheduler regulates when it will be transmitted to the recipient UE.

- **User Equipment (UE):** The mobile device of the user. It can be any device with mobile communication capabilities. In our scenarios it is a smartphone.

Different types of traffic have different requirements in regards to latency and guaranteed bitrate. Thus, in order to maximize QoS, the LTE architecture is able to treat some data differently. LTE uses the concept of bearers to route IP traffic from a gateway to the UE. A bearer is defined as: " an IP packet flow with a defined QoS between the gateway and the UE" [34]. When a device is attached to the network a default bearer is created, that is able to carry all types of traffic without any QoS provisioning. When the network is required to carry traffic with QoS requirements a dedicated bearer is set up. The dedicated bearer exists in parallel with other dedicated bearers and the default bearer. It serves as tunnel to carry this traffic, and uses a Traffic Flow Template (TFT) to specify the QoS provisioning (*e.g.*, delay, guaranteed bitrate) applied to the traffic. In practice though, only VoLTE traffic is served by a dedicated bearer. To the best of our knowledge, all other traffic is served by the default bearer. Thus, in the sequel the algorithms we propose may treat all multimedia and web traffic as equal.

The UE connects to the operator network through any of the multiple base stations (eNodeB) that the operator controls, as shown in Figure 2.1. EnodeBs are in turn connected to the core network (CN) of the operator. This set of enodeBs can be collectively called Radio Access Network (RAN). They form the interface between the UE and the operator.

When a mobile user surfs the web, a TCP connection is established with a remote server. The packets associated with this TCP connection travel through the Internet, enter the core network of the mobile operator through the PGW, and are then routed to the serving base station (eNodeB) that the UE is connected to. At the eNodeB, the packets are stored in a buffer dedicated to the target device. The allocation of resources to the connected UEs is determined by a scheduling mechanism.The packet remains in the dedicated buffer until the scheduler decides to allocate resources to the recipient UE.

The scheduling decision is taken periodically, once every Transmission Time Interval (TTI). This period largely differs among mobile telecommunication systems, with more recent technologies having lower values. The TTI of the downlink for the Frequency-Division Duplexing (FDD) version of LTE, which is used by the majority of operators, is fixed to 1 ms [35]. For the Time-Division Duplexing (TDD) version, used mostly by operators in China [36], the TTI is in the range of a few ms, depending on operator configuration. For the older UMTS technology this value is at least 10 ms. Thus, the UEs receive data in a way such that a burst of data is transmitted to them, during TTIs in which they have been allocated resources and receive nothing during TTIs in which they have not been allocated resources. The scheduling process is usually based on a fairness scheme that takes into account the data requirements and channel quality of all the UEs served by the same BS. A very popular such scheme is the "proportionally fair" scheduling [37].

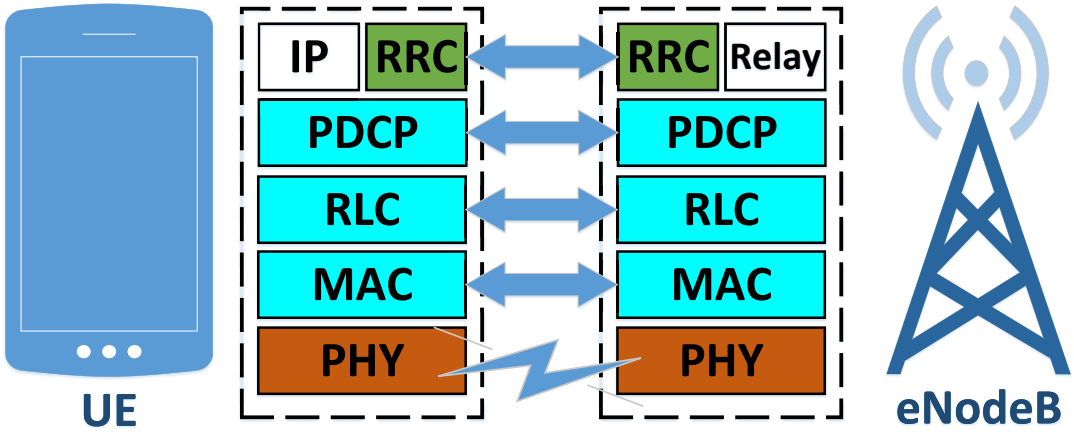


Figure 2.3: The LTE protocol stack. Protocols of the same layer have the same color.

It tries to weight the past allocation of resources and the current potential throughput of all the competing users. This way it finds a balance between providing adequate resources to all users, regardless of their channel quality, and maximizing the overall throughput of the base station. Thus, in contrast to wired networks, which usually serve traffic based on a FIFO scheme, the incoming traffic at the antenna is distributed to user specific queues and the outgoing is shaped by the scheduler. So, the nature of the competing traffic (UDP/TCP or short/long flows) does not greatly affect the speed of each user. On the other hand, factors that may have an effect include policies (e.g., whether a user is a virtual or host network subscriber[1] [38]) and the specific service that generates the traffic (e.g., VoLTE traffic has the highest priority in an LTE network).

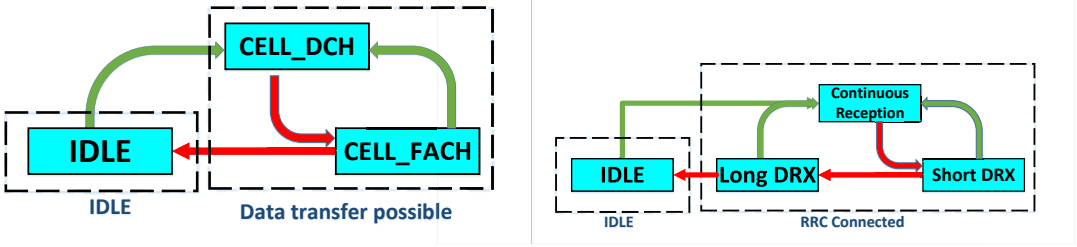
When a UE is scheduled, the packets present in the buffer are grouped into a Transport Block (TB), which is then sent to the UE. In case of a very bad signal and/or a small amount of allocated resources, just a segment of a packet can be encapsulated in a TB.

2.3.2. The Lower Layers of LTE

The amount of bits that may fit in the TB is a function of the amount of radio resources allocated by the scheduler in a specific TTI and the Modulation and Coding Scheme (MCS). In LTE, radio frequencies are organized in groups of 14, called Resource Blocks (RBs) and RBs in turn are organized in groups of varying size (2, 3 or 4), forming a unit called Resource Block Group (RBG). RBGs are the quantum of resources that can be allocated to the users. The amount of bits that each RBG can "carry" depends on the MCS used for it. The UE periodically notifies its signal quality to the antenna, which then decides the MCS value for the transmission.

The physical layer of the UE receives the TB and the LTE protocol stack starts processing it. Figure 2.3 presents the main layers of the stack. In brief, the layer 2 of LTE is composed of 3 sublayers managing the data transmission, on top of which is another layer responsible for the

[1]It is documented that users of virtual operators have lower priority to the users of the corresponding host operator.



(a) 3G (b) 4G-LTE

Figure 2.4: The RRC state machines of the 3G and 4G-LTE mobile communication standards.

signaling between the UE and the base station. A bottom up presentation of these sublayers is the following:

- **Medium Access Control (MAC):** The first layer above the physical channel is responsible for, among others, the integrity check of the TBs. If it detects errors in the TBs it requests a fast retransmission of the problematic TB, a process called Hybrid Automatic Repeat Request (HARQ). A retransmission may happen no sooner than 5 ms after the original transmission, even though it has priority over normal transmissions. The MAC layer stores all the received versions of the TB and then applies operations to them in order to recreate the original TB.

- **Radio Link Control (RLC):** This layer receives TBs and uses them to reconstruct packets of the upper sublayer (PDCP). RLC packets have a unique sequence number that ensures in order delivery to the receiver. If RLC detects that a packet is missing, it stops forwarding packets to the upper layers until it is received. While it waits for the missing RLC packet to arrive, the incoming packets that have a higher sequence number than the missing one are stored in a buffer. If the missing packet does not arrive within $35 - 100$ ms, it is considered lost and requested as new data. The RLC packets with smaller sequence number than the missing packet are then delivered to the upper layer. Such events may add significant inaccuracies to applications monitoring traffic and are taken into account in the algorithms we develop in the sequel.

- **Packet Data Convergence Protocol (PDCP):** In general, a PDCP packet corresponds to an IP packet. This sublayer prepares IP packets for transmission over LTE. For example, it may apply header compression, by replacing the IP header with a token that has a maximum size of 4 bytes, effectively reducing the data needing transmission.

The Radio Resource Control (RRC) layer is responsible for exchanging connection related information between the UE and the antenna and is not directly involved in the data transmission. It is on the same layer as IP. Operating the radio is very costly both in terms of power and network resources. The radio is only activated in periods when data transmissions are expected. The radio

turns on to handle a transmission and then remains on in anticipation of more future transmissions until a timer expires. This state is called "continuous reception". If the timer expires before any new transmissions take place, the radio enters Discontinuous Reception (DRX), where it sleeps for a portion of the time, while maintaining the established radio context (*e.g.*, remains synchronized with the antenna). The DRX state is split into two phases where the proportion of time spent sleeping, is related to the anticipation of a transmission. If it remains in this state for long enough without any transmission it returns to the idle state. The timers that determine how much time the radio remains at each state are called "RRC state machine". A graphical representation of the transitioning between the states is shown in Figure 2.4b. The RRC timers' exact values are operator dependent. Changing state is a very time consuming process. The RRC state machine adds extra delay only to the first packet of a connection or to packets of connections with very infrequent transmissions though.

Please note that the RRC state machine of 3G technologies is different, as can be seen in Figure 2.4a. In brief, there is no DRX phase, but between the connected and the idle states there is another one which consumes about 40% of the energy and allows the device to access a low speed (about 20 Kbps) shared channel.

2.3.3. Baseband

There is very limited public information on the lower layers of mobile hardware. Also, the experts in this area are mostly found in companies rather than academia. In the following paragraphs, we will attempt to present an overview of what is known to be common practices of the major vendors, regarding the low level design of their communication hardware. Smartphones intergrate a lot of their hardware functionality in a System on Chip (SoC). Among others, a SoC contains a cellular modem for connecting to mobile networks and a Wireless Local Area Network (WLAN) module for connecting to WiFi access points. The code of the core OS[2] and most of the device's drivers is executed by a part of the SoC called Application Processor (AP). All the radio devices of a phone are controlled by a part of the SoC called Communication Processor (CP), or just modem. In older devices the CP used to be a separate chip, but nowadays it is usually integrated in the SoC. The reasons for separating the applications and the communications functionalities of the device include the increased complexity of the communications code and isolating low level communications from applications. Both AP and CP can write and read a region or all of the main system memory in a process called Direct Memory Access (DMA) and this is how they communicate[3].

The code executed by the CP is independent of the core OS and is essentially a secondary OS inside the device. This code, the firmware of the CP, is called baseband. It is notoriously hard to find public documentation of its operation, to the point that there is speculation that not even

[2] In this thesis this is always Android.

[3] There is a set of commands called "Hayes command set", dictating how the two processors exchange information, mostly related to handling voice and signaling. This is out of scope of this thesis though.

smartphone manufacturers have access to parts of it. It is suspected that CPs are very vulnerable to exploits, so the very limited number of vendors capable of building them, release as little information as possible. As a consequence, in the experiments we will discuss in the sequel, this part is treated as a black box. We can have a clear view of the functionality of the main OS, but can only assume that the baseband is following the 3GPP specifications. 3GPP specifications though are very complex and allow in a lot of cases functionality to be implemented differently by the various vendors. In some of the experiments, we resorted to monitoring eNodeB control messages, in order to get insights about how the lower layers of the mobile networking stacks operate on the smartphones we examined.

2.4. Linux Kernel Networking

In this section, we will attempt to shed some light on how a packet moves up in the kernel networking stack. It is said that Linux Kernel Networking is a "sealed garden", despite being the most famous open source project. The related code is very dense [39] and has very few comments. Public documentation and tutorials are hard to find, outdated and to a big degree inconsistent. Due to number of companies involved in building Anrdoid phones, the used kernels are usually very behind the upstream versions, and thus lack some modern features. The main deciding factor for picking a kernel is based on what the SoC supports and SoC vendors tend to not provide updates after a chip is on the market. Before diving into how Linux and, by extension, Android handle packet transmission and reception and how it interacts with the NIC, we have to define some related concepts.

DMA is a feature of modern hardware, which allows a component of a computer to write directly to the main system memory. In the case of networking, before DMA the packet would have to be received by the NIC and then an interrupt would be sent to the CPU to handle the packet, a process with a great impact on the efficiency of the CPU. With DMA, the NIC may write the incoming packets as they arrive to a designated area in the main memory of the system, without interfering with the normal CPU operations. Since the first stages of packet arrival are independent of the CPU, it is possible to disable network interrupts, without dropping packets. This allows for efficient handling of big numbers of packets without choking the CPU.

The ring is a circular data structure of fixed size. A major benefit of a circular buffer is allowing different processes to add and remove elements at the same time, without conflicts, thus enabling asynchronous operation. A ring buffer is used by network cards to store references to packet descriptors. For example, in the case of packet transmission the operating system would put data in the ring and then move a pointer indicating what is the last element added to the ring. There is another pointer of the network interface indicating the last element the network interface transmitted. As the interface is transmitting packets it moves its pointer until the element pointed by the pointer of the OS, at which point there would be no more data to transmit.

The contents of the ring are fixed size pointers to data structures called Socket Buffer Struc-

tures (SKBs). The Socket Buffer Structure (SKB) is the most basic data structure in the Linux networking code. Every packet sent or received is associated with at least one SKB, which holds packet related metadata, headers of the various layers, layer specific fields and part or all of the actual packet. Through this structure the different layers are coordinating their processing of each packet. Upon booting the device, the kernel informs, through the related drivers, the network interfaces which area of the main memory is reserved for storing packets and initializes a number of SKBs there. After a packet fully arrives at the NIC and the NIC finishes processing it, it is written in the specified area of the main system memory through DMA. Writing the packets directly at the main memory, removes the need for coping them as they are processed by the network stack, resulting in faster processing. The DMA process checks the ring to find an unused SKB and copies the payload and headers to the related fields of the structure. It also sets up some flags in the SKB structure indicating the processing the packet has already received and packet metadata. If the memory dedicated to packet handling is full, the packet is dropped without consuming any CPU resources for processing and the hardware issues an error. If the packet is too big to fit in a single SKB, multiple descriptors are fetched from the ring and the contents of the packet are split among the related structures they point to. Then a hardware interrupt is sent to the CPU, notifying of the arrival of the new packet. The exact handling of the interrupt will be discussed later. Upon deciding to act on the packet, the OS reads the packet data and sends it to the TCP/IP stack for further processing. Once the associated SKBs and descriptors are no longer in use the OS releases them.

Code 2.1: Some important fields of the `sk_buff` structure.

```
struct sk_buff {
  /* These two members must be first. */
  struct sk_buff *next;
  struct sk_buff *prev;
  struct sk_buff_head *list
  struct sock *sk  // The socket that owns this buffer
  ...
  union {tcphdr; udphdr; ...} h;     // Transport header
  union {iph; ipv6h; arph; ...} nh; // Network header
  union {raw} mac;                   // MAC header
  ...                                // Data
}
```

Part of this structure's definition in the Linux kernel is shown in code snippet 2.1 [40]. The first fields are related with the data-structure itself, which since it is a version of a doubly linked list, it needs pointers to the next and previous elements. In order for every entry to be able to find the first element quickly, there is another pointer to a dummy structure (`sk_buff_head`), placed

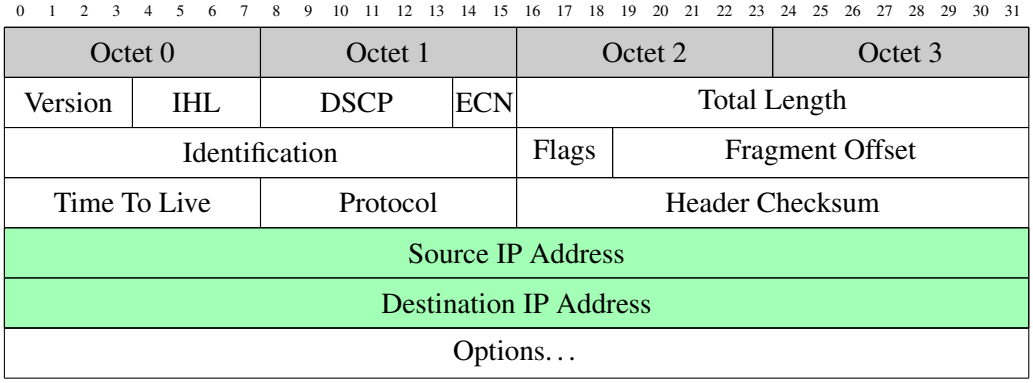


Figure 2.5: The fields of a IPv4 header.

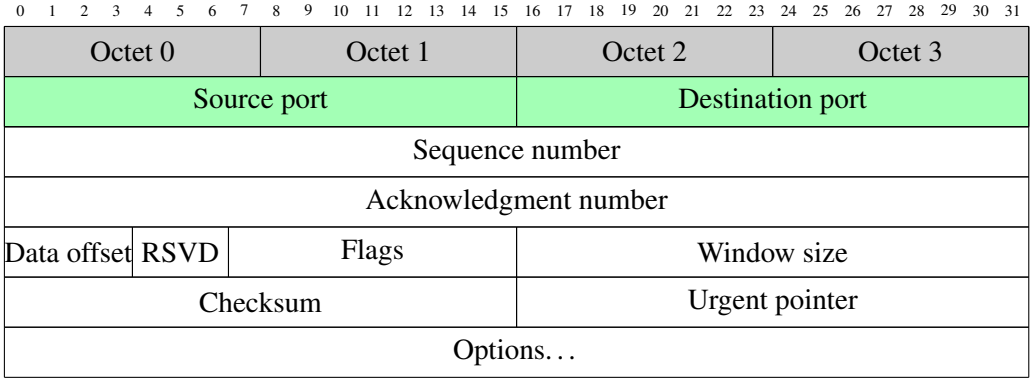


Figure 2.6: The fields of a TCP header.

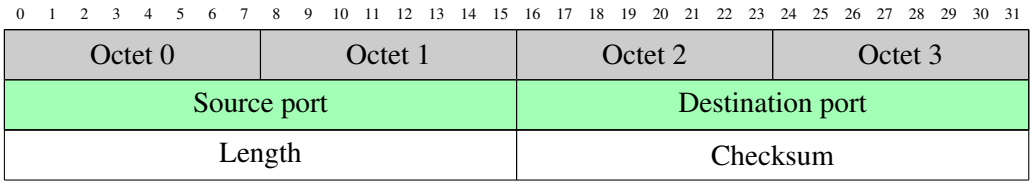


Figure 2.7: The fields of a UDP header.

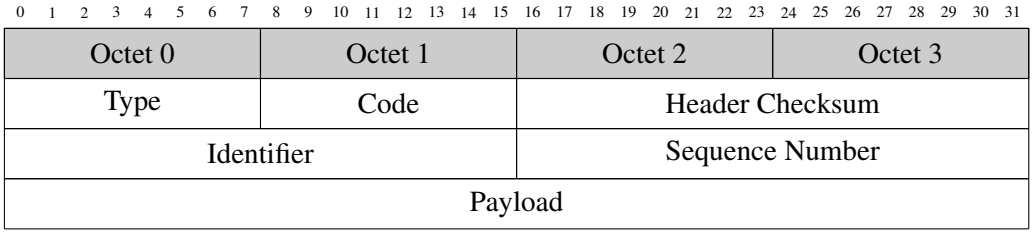


Figure 2.8: The fields of a ICMP echo request / reply header.

at the head of the list. This dummy structure holds information about the first real element of the list and the total number of elements and controls concurrent access to the list with lock fields. SKB is modified based on the processing done to the packet on each layer. For example, an outgoing packet will have IP header related entries added at the IP layer. It is tied to a socket through a pointer to a `struct sock`, a structure holding networking information for sockets [41].

The lower part of code snippet 2.1 presents some unions, which hold various structures related to the different networking layers. A bit indicates which process has control over each ring entry. There is a pointer marking the exact location of the data in the packet. In the case of an incoming packet, it points to the packet as it was received. As the packet moves up in the stack, each layer adds a pointer to this layer's header and moves the data pointer after the end of its header. In case, the packet is outgoing, every layer writes its own header initializing the related pointer.

In systems with multicore processors, incoming packets are spread among the CPU cores, in order to among others balance the load. Each CPU core has its own receive queue (rx_ring) and when an incoming packet generates an interrupt it is possible to be directed to that core (*e.g.*, each queue has its own interrupt number). The packets are split among the queues based on hashing parts of their headers. For example, if the received packet is TCP or UDP, the location of the bits related to the IP addresses and ports are fixed. This can be observed in their headers shown in Figures 2.5, 2.6 and 2.7, where the related fields are marked in green. Thus, the hardware may hash these specific bit ranges, then divide the results by the number of rings and assign the packet to the ring with number equal to the remainder of the division. This process guarantees that packets of the same TCP / UDP flow will be assigned to the same ring, processed by the same core and therefore arrive at the higher layers in order. This mechnism is called in the documentation of the Linux Kernel: "RSS: Receive Side Scaling" [42]. The exact hashing procedure varies among hardware vendors, but the results are similar. For example, even if only the IP addresses are hashed, it is enough for the packets of the same flow to be assigned to the same ring. Curiously, the field that defines a flow in the case of ICMP echo messages, namely the "Identifier" field, is in a different bit range compared to the bit ranges of port numbers in TCP and UDP headers, as it is shown in Figure 2.8. Thus, it is not guaranteed that these packets will always be assigned to the same ring.

Timestamping packets is an expensive operation, so it is enabled only when it is needed. Since timestamps are crucial for traffic analysis they are enabled when packet capture software such as tcpdump is in use. The timestamps represent either the time the packet was sent or arrived. For arriving packets specifically, the timestamp is added right after the packet moves out of the link layer. In the case of the LTE stack, the last part of the link layer is the PDCP sublayer. The timestamps are stored in the SKB of each packet.

2.4.1. Interrupt Handling

There are two ways for the OS to be notified of a new packet: interrupts and polling. Interrupts offer the minimum possible receive latency per packet, at the cost of high processing overhead.

Modern CPUs have features such as long pipelines and speculative executions, making interrupt handling very costly and thus not suitable in cases of frequent packet arrivals. Polling allows the CPU to check for packet arrivals at specific intervals, reducing the CPU cost, but increases delay and jitter between packet arrival and processing. Interrupts are better in cases of infrequent and unpredictable arrivals. On the other hand, polling is better in cases of high volumes of traffic. The polling frequency should balance the tradeoff between overhead and delay. When a new packet arrives the NIC always issues a hardware interrupt, which is received by the driver. It is up to the driver when to send a software interrupt to the kernel and by extension the CPU. In order to increase efficiency, interrupts are disabled while the CPU is processing packets and each receive queue is independent, so the related core can work on its backlog safely.

It is possible that the CPU is not able to handle all the incoming packets during a busy period, either because a very big burst of packets arrives at once or because of competition to CPU resources from other processes. The CPU relies on buffering the packets it cannot process right away, but if packets continue to arrive at a higher rate than they can be processed, some packets will eventually be dropped.

The default behavior of the Linux kernel is to rely on interrupts, but if the flow of incoming packets increases beyond a certain threshold, it switches to polling. This behavior is dictated by an extension to the packet processing framework, designed to improve the performance of high-speed networking called New API (NAPI). Upon switching, the first polling request takes place as soon as the last interrupt related packet is processed, since it is assumed that a flow of packets is incoming. While NAPI polling is active, the kernel checks for new packets at fixed intervals or when the NIC receives a specific number of packets or when an application requests for network data. When the number of unprocessed packets falls below a threshold, the kernel switches to interrupts again.

As of writing this thesis, the Linux kernel supports busy polling [43]. Busy polling reduces interrupt and polling latencies at the cost of consuming CPU cycles. The developers of this feature describe it as: “the ability for a user thread waiting for some incoming network message to directly poll the device ring buffer.” [44]. The CPU is not occupied by another process in between arrivals, thus context switching delay is avoided and the CPU can act on the freshly arrived packets as soon as possible. Thus, one CPU core is dedicated to waiting arrivals of a specific process, improving significantly the throughput and latency parameters. It is up to the application developers whether to use this functionality though. To the best of my knowledge, busy polling is not yet included in Android. For the time being it is aimed towards datacenters, where each machine may have upwards of 100 cores. However, modern mobile CPUs are increasing their core count and we expect future mobile CPU implementations to support this feature. Once Android supports it, it is expected to reduce the effects of various measurement artifacts described in later chapters of this thesis, therefore increasing the accuracy of the presented techniques.

2.4.2. The Higher Layers of the Linux Networking Stack

Once a packet has been received by the kernel it is sent to the queuing discipline (QDisc) layer. This layer is between the NIC and the IP stack. It is responsible for enforcing traffic control in the Linux kernel. The capabilities of the QDisc layer in manipulating outgoing traffic are: "shaping" (*e.g.*, smoothing big bursts of traffic and lowering available bandwidth in general) and "scheduling" (*e.g.*, reordering packets in order to protect interactive traffic from bulk downloads), When the traffic is incoming, it applies "policing", which has the same main objectives as "scheduling" but in the ingress. Finally it also applies "dropping" in both directions, in cases where traffic exceeds a set bandwidth. It should be noted that mainstream Android distributions have very limited support for QDisc, whereas some more specialized, like Cynaogenmod, support its full functionality.

Then, the IP stack further processes the packet as it goes up the network layers. Eventually, the payload of the packet will be delivered to the corresponding socket of the application requesting the data. The only interesting aspect in this part of the packet path in regard to delay is whether the socket is "blocking". When an application thread makes a call to a blocking socket, in case the request cannot be fulfilled right away, the thread will wait until the socket can fulfill it. Such cases include when a thread tries to push data to a full socket buffer or read data from an empty socket buffer. In contrast, if a socket is "non-blocking", the call will return immediately with an error message and it is expected that the application will make another call in the future. Therefore, blocking sockets have the smallest added delay in data transmission, with the price of forcing threads to wait. This option is important in case we want to run delay sensitive algorithms at the application level. The default behavior for TCP sockets is "blocking".

One last thing we have to mention is under which conditions the OS is forced to make a DNS request, when an application wants to map a domain to an IP address. Initially the program making the request will check if it has the IP of the target host is already stored in its cache. If not, it will call the OS function `gethostbyname` to do the resolution. If the OS has the address stored in its own cache or there is mapping of the host to an IP in a special file called `hosts`, it returns the IP to the program. If not, the OS will make a DNS request to the preconfigured DNS resolver. The default DNS resolver is usually set by the ISP to be one of their own servers. Users may modify the DNS resolver to any of the publicly available DNS services, such as Google public DNS, Cloudflare and OpenDNS. In later chapters, we investigate the implications of choosing a DNS server other than the default offered by the ISP.

2.5. Final Notes

In this last section, we mention some other networking aspects of smartphones that can improve the context of this thesis, but did not fit in any of the sections above.

In this thesis, we assume that the network configuration of the mobile devices is static. So, if a mobile is using the mobile interface, it will continue using it with the same IP. In normal

operation this might not always be the case. While a connection is active another interface might come up (*i.e.*, a download was started over mobile and the WiFi comes up in the meantime) or IPv6 autoconfiguration might change the IP address of an interface (*i.e.*, use of tentative IP address).

The data usage tracking on an Android device is implemented by counting bytes per application - User Identifier (UID) / network interface combination. The results are outputted in the `\proc` directory. The operators track the data usage of specific users[4] based on the number of packets that pass through the PGW in both directions. Since these are two different vantage points, it is common to report different values. This is pronounced in the case of high bitrate UDP traffic, which might overflow buffers on the path between the PGW and the UE.

Android is able to perform "per application routing / socket routing". Users might want to connect at the same time to the Internet and a device like a goPro camera, a drone or a wireless printer. Thus, the mobile interface should provide Internet access to all the other processes, while the WiFi interface is connected to these devices. Incoming connections are mapped to network interfaces based on which interface received the SYN packet.

[4]A user is identified as an International Mobile Subscriber Identity (IMSI), a unique number included in the Subscriber Identification Module (SIM) card.

Chapter 3

Related Work

In this chapter, we review the state of the art literature and emphasize the differences with the sequel of this thesis. We start with presenting some characteristic examples of bandwidth measurement approaches, which vary between active, lightweight active and passive in Section 3.1. Then Section 3.2 gives an overview of past work on the analysis and evaluation of mobile communications standards and finally, Section 3.3 summarizes studies evaluating the performance of third-party services.

3.1. Mobile Bandwidth / Capacity Estimation

3.1.1. Active Measurement Techniques

Nowadays, the most popular solution for mobile bandwidth estimation is Ookla's mobile application *Speedtest* [45], which makes use of active measurement techniques. To estimate the bandwidth available to a mobile device, Speedtest tries to saturate the device's downlink by downloading a large file through two parallel long-lived TCP connections in a test that lasts for about 10 seconds. Speedtest can connect to many well-connected measurement servers deployed by Ookla, so the measured bandwidth is less likely to be affected by cross traffic. To the best of our knowledge, there is no public documentation detailing the algorithm used by Speedtest on mobile devices. Our understanding is that every second, a certain amount of traffic samples are generated. These samples are aggregated into 20 bins, with each bin containing about 5% of the samples. The bins are then filtered to remove measurement artifacts. The final estimation is calculated by averaging over the bandwidth values of the remaining bins.

A similar approach is proposed by [15], where 3 parallel TCP connections are established with the three closest servers to reduce the impact of TCP's receive window, overloaded servers, and packet losses. The traffic samples are segmented into equally sized bins and samples collected during the slow-start phase are discarded. In order to reduce the effect of outliers, the median of the bandwidth estimated in the remaining bins is taken as the final estimation.

In [46], authors calculate the end-to-end throughput availability by sending high rate UDP

traffic, while taking into consideration packet interarrival times and the scheduling effects of mobile networks.

These tools may provide the most accurate bandwidth estimation, but are unsuitable for frequent usage on mobile devices, which is the ultimate goal of our tools. They rely on transfer of large amounts of data over prolonged periods of time and thus, incur a high overhead.

3.1.2. Lightweight Active Measurement Techniques - Packet Dispersion

"Packet dispersion" is a lightweight active measurement technique. Packet pairs or packet trains are transmitted from a server to a target device, which timestamps their arrivals. It is meant to measure the asymptotic capacity of the path's bottleneck link by analysing the time dispersion of packet arrivals. It is possible to estimate the capacity of the bottleneck link in the path, based on the fact that the average throughput measured by packet trains converges to the asymptotic dispersion rate, from which an estimate of the bottleneck capacity can be computed.

Lai [47] attempts to actively measure the link capacity (which in [47] is called bandwidth) of a path by taking advantage of the packet pair property of FIFO-queuing networks. Dovrolis [48] further refines the packet pair technique and demonstrates that packet pair dispersion rate has a multimodal distribution, whose modes in turn depend on the capacity and the cross traffic at each of the links composing the sender-receiver path.

CapProbe [49] proposed a technique based on packet pairs dispersion and delays to devise a reliable capacity estimation technique, aimed at mobile networks. This approach though is very sensitive to topology parameters, like the number of nodes in the path and the link utilization. The number of packet pairs required to generate a valid estimation rapidly increases, when these parameters increase. Furthermore, it ignores the effects of the mobile scheduler.

Packet dispersion offers a good estimate of the capacity, while generating very little traffic. As we will show in Chapter 5, applying packet dispersion techniques in a mobile network is problematic, because the scheduler of the base station either shrinks or enlarges the dispersion greatly. Also, CapProbe is not robust enough to be used in most scenarios. Finally, these techniques are meant to measure the capacity of the bottleneck link of a path. Instead, we are interested in measuring the per user capacity of the cell at a given moment. Only the latter metric may by used by bandwidth optimization algorithms.

3.1.3. Passive Measurement Techniques

Conversely, passive monitoring techniques aim at estimating similar information by analysing ongoing mobile communications, without triggering any dedicated activity. Gerber *et al.* [50] achieved quite accurate results just by relying on selected types of applications (i.e., video streaming), which provide more reliable throughput measurements as they are more likely to exploit the full cell capacity. Their method first sniffs all the traffic going through a certain vantage point inside an operator's network. Then it identifies the traffic flows that belong to applications that

are not rate-limited at the server side and are guaranteed to use all the resources that the base station allocates to a user.

In order to study transport protocols in LTE, [16] developed a passive measurement scheme, which monitors the sending rate over a given time window that ensures the full exploitation of the capacity. PROTEUS [51] combines passive monitoring with linear prediction to estimate the achievable throughput. Other solutions worth mentioning in this category are [52], where the authors try to identify bottleneck links in the core network of an UMTS operator by conducting large scale passive measurements of TCP performance parameters and [53], where network "footprints" (generated by counting the number of packets and the number of retransmissions of all the users of a network) were used to identify capacity bottlenecks.

The above solve the major problem of generating traffic, a resource very limited in a mobile scenario. However, they cannot be directly applied to mobile phones. It is important for traffic prediction algorithms to be aware of the context of specific users. Some of these techniques are implemented inside the operator's network. Therefore, they are not applicable at the application level and require the cooperation of the operator.

We conclude that none of the aforementioned solutions allow for frequent throughput measurements, nor do they provide estimates of the per user cell capacity on the client side (mobile device) to allow for effective capacity prediction and resource allocation. In Chapter 6 we propose a passive technique that is able to provide an estimation of the per user capacity range by monitoring the packet arrival patterns that take place during the TCP slow start phase. In this thesis, we are interested in a more accurate per user capacity measurement that is based on periodic samples of the exchanged traffic, taken during the whole duration of the flow.

3.2. Mobile Communication Standards Mechanics and Measurements

None of the works presented in Section 3.1 attempts to validate the accuracy of their measurements when these are performed directly on the smartphone. A high guarantee of measurement accuracy is critical though for bandwidth prediction algorithms, such as the ones presented in Appendix A. To this end, Chapter 4 is dedicated to using accurate LTE scheduling information in order to validate the measurement algorithms presented in Chapters 6 and 7. In this section, we present papers which take measurement accuracy into account by comparing their results to some kind of baseline.

Huang *et al.* [16] studies LTE performance by measuring mobile phone data. To validate their findings, the authors performed experiments with controlled traffic patterns, which served as a reference point.

The authors of [54] use a real-time spectrum analyser to detect the amount of LTE resources M2M devices utilize and subsequently evaluate the performance of channel aware algorithms.

Xie *et al.* [55] use an energy-based spectrum monitor to assess the the average fraction of used

RBs. Even though this is a robust solution, we are interested in more fine grained control over the resources specific users use. The same team has proposed another approach to monitor LTE control channel scheduling information [56]. However, they admit that it is reliable only when the wireless channel has also zero error and it does not scale well with an increasing number of users. In this thesis, we use a solution developed in house at IMDEA Networks Institute. OWL [20] is a reliable LTE control channel sniffer. It is able to accurately decode more than 99% of LTE the control messages. That way we can obtain a virtually complete log of base station scheduling.

MobileInsight [57] is able to access the debug log of the radio chipset of phones with Qualcom SoC. It is the solution with the least overhead for monitoring physical layer traffic, but it is limited to a single vendor, whereas in Chapter 4 we are interested in studying the behavior of phones with different vendors of SoC. Thus in the sequel, we rely on OWL for trustworthy radio level information. It is has complete information of all the smartphones connected in the monitored cell, in contrast to the aggregated information that power-analysis based solutions provide.

Li *et al.* [58] assess the measurement accuracy of smartphones over WiFi. In their testbed they use three laptops, which serve as the ground truth, to sniff the traffic a smartphone is receiving. Then, they compare when the related packets appear in tcpdump traces collected on the smartphone and the laptops. In our experiments though, we observe that sometimes LTE and WiFi have a drastically different behavior at the same device, prohibiting us from using this solution.

A few papers [59–61] use commercial tools and/or operator network information to evaluate LTE performance. For instance, direct access to network logs is used in [59] to provide a detailed comparison between LTE and UMTS, but since network logs are usually always anonymized (if they are released at all), it is impossible to identify a given device under test among the set of traces. A commercial LTE modem from Teldat is used in [60] to measure application level performance. Similarly, a network maintenance tool is used in [61] to analyse quality of service, but such tools only provide compound information averaged over periods of time, which do not achieve the required granularity for our measurements.

3.3. Characterization and Evaluation of Cloud Service Providers

CSPs' performance in mobile networks may be affected by multiple factors such as radio link variability, the presence of in-path middleboxes [62], traffic shaping policies [63, 64], the behavior of the DNS resolver [65, 66], the peering relationships between cloud providers and MNOs [67, 68], and inflated network paths [38]. While previous research studies assumed that users are always paired with geographically close content replicas thanks to DNS-based geolocation techniques [69] and IP anycast, [70] showed that this assumption might not always be true due to inaccurate geolocation of mobile users resulting in sub-optimal server assignment. Krishnan *et al.* (2009) [71] demonstrated that redirecting clients to the server of a CSP with least latency does not guarantee an optimum client latency.

Furthermore, Goel *et al.* [72] investigated how content served by third-party service providers

for a certain web pages might impact the end-user's experience while browsing. The authors report third parties inflate the page load time by as much as 50% in the extreme case. Rula *et al.* conducted a crowd-sourced measurement campaign to study DNS behavior in mobile networks and revealed that client-to-resolver inconsistencies make DNS-based solutions unsuitable for determining the location of clients in MNOs [65]. Recently, the Internet has witnessed the growth of new anycast-enabled CDNs (*e.g.*, CloudFlare) to overcome these limitations. However, this strategy depends on the stability of the paths toward the nearest server.

Part II

Smartphone Measurements on the Physical and Lower Layers

Chapter 4

LTE Radio Link Estimation Accuracy of Smartphones

This chapter addresses the accuracy of LTE performance measurements on mobile phones, such as the ones proposed in the next chapters. More specifically, we study whether mobile phones can accurately measure LTE radio link data rate. We do so by comparing ordinary smartphone measurements, performed at the kernel and application level, to the actual LTE scheduling information obtained through our reliable LTE control channel sniffer: OWL [20]. These physical layer LTE measurements serve as the ground truth for the measurements of the higher layers.

We have developed software components to generate data traffic between a "remote" server and a UE, using three different smarphones. The generated traffic is monitored from a total of five vantage points, located in various networking layers of three physical locations. Using a variety of phones allows us to examine whether different chipsets exhibit variations in the reported accuracy.

We perform two measurement campaigns. The first is focused on attempting to quantify the latencies that the different layers of the phone add in the transmission / reception process. Studying these latencies we can reveal the source of inaccuracy in link rate estimation. The second campaign compares data rate estimates as they are reported by monitoring traffic at different layers.

In summary, our experiments report that the mobile phones are able to perform accurate and precise data rate measurements, even when the measurement is performed over small data bursts. The accuracy between the uplink and the downlink is different though, due to the different processing followed in each direction, with the uplink being less accurate. Finally, we observe measurable differences in the biases different phones add to the estimation.

The rest of the chapter specifies the measurement setup and the devices involved in Section 4.1 and Section 4.2 discusses the two measurement campaigns and our findings. Section 4.3 concludes the chapter.

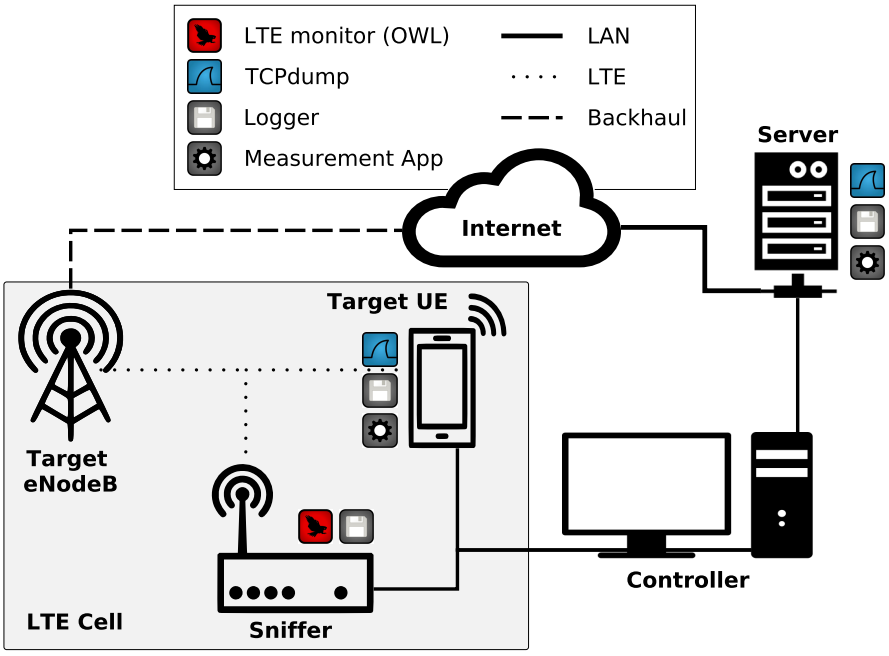


Figure 4.1: Experiment setup showing devices, connections and software (figure from [1]).

4.1. Testbed

Figure 4.1 illustrates our testbed as well as the vantage points from where we monitor traffic. The following steps serve as a walkthough of the downlink scenario, where a target device receives traffic from a remote server. The uplink scenario is similar, but for some small differentiations in the physical layer of LTE, which will be discussed later in Section 4.2. We use a well connected university server to transmit traffic towards the phone being measured. The traffic travels through the Internet and then through the mobile operator's network until it reaches the eNodeB that serves the phone. Before the traffic is transmitted, the eNodeB uses the control channel of LTE to coordinate with the phone the TTIs during which the transmission will take place. OWL, the LTE sniffer we use, intercepts these unencrypted control messages and logs them for offline processing. The actual data transmission follows, where the IP packets the server had originally transmitted, are sent from the eNodeB to the UE encapsulated in TBs as we discussed in Chapter 2. OWL is able to sense power on the channel during the data transmission, but this transmission is encrypted. Since, relying on detecting power makes it impossible to distinguish our target device from other devices using the same cell, we ignore the actual data transmission at the physical layer. We rely only on the unencrypted scheduling information. Finally, the UE receives the traffic, and forwards the related packets to the upper layers of the networking stack and eventually to the application requesting them.

The vantage points where we monitor traffic are the following. At the server and the phones we use tcpdump to monitor IP packets at the kernel and we also log socket events as they are

Table 4.1: Technical specifications of the test phones.

Phone	Motorola MotoG 4G (2014)	Huawei P8 lite (2015)	ZTE Blade A452 (2015)
Chipset	Qualcomm Snapdragon 400 MSM8926	Huawei HiSilicon KIRIN 620	MediaTek MT6735P
CPU	ARM Cortex-A7 1200 MHz (4 cores)	ARM Cortex-A53 1200 MHz (8 cores)	ARM Cortex-A53 1000 MHz (4 cores)
OS	Android: 4.4.2 KitKat	Android: 5.0.2 Lollipop	Android: 5.1 Lollipop
RAM	1 GB	2 GB	1 GB

reported by the application sending or receiving the data. The sniffer logs the physical layer between the UE and the eNodeB at both directions. At each layer we report the data (*e.g.*, TB, IP packet, socket.receive, socket.send) size and the related timestamp. After each experiment we combine the log files offline to identify the same packet transmission at each vantage point.

It is important to use a well connected server, because any cross traffic in the Internet may distort the packet transmission patterns we use. As we will discuss in the sequel we rely on these patterns to identify specific transmissions. Further, all the measurements are performed at a cell which does not have much congestion (*i.e.*, usually less than $10 - 20\%$ of the available RBs are used by competing traffic) and the connected phones have good signal quality. This further helps us identify specific packet transmissions.

The scheduling information the eNodeB transmits identifies the UEs based on their Cell Radio Network Temporary Identifier (C-RNTI). We are not able to extract the C-RNTI from the phones. Thus, in order to identify the C-RNTI that belongs to our target device, before each experiment we transmit a specific sequence of packets and we look in the sniffer logs which C-RNTI has a similar pattern. This C-RNTI remains valid for the rest of the experiment, but resets soon after we stop sending traffic.

We use a variety of phones which belong in the same category, performance and price wise, but use different chipsets and Android versions. Our goal is to examine whether the manufacturer and OS version affects accuracy. The exact specifications of each test device are reported in Table 4.1.

The above components are synchronized by the controller PC, which coordinates the experiment. The controller manages the server through an ssh session and is directly connected to the target UE and OWL over USB. At the end of each transmission collects and combines the logs.

It should be noted that we try to have the various devices as much in sync as possible, but we are not able to achieve a synchronization of less than a few hundred ms. As we will discuss in Section 4.2, the experiments are formulated with this fact in mind though and thus are not affected.

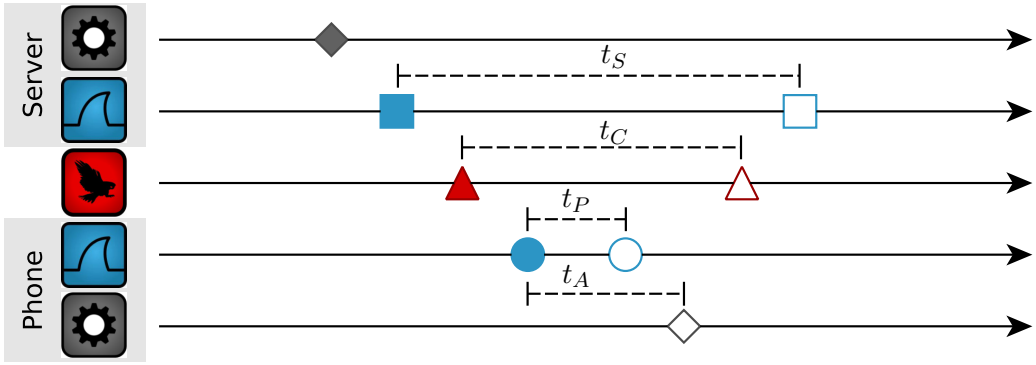


Figure 4.2: Communication diagram for the downlink isolated transmission. Dimension lines illustrate data-to-ack latency (figure from [1]).

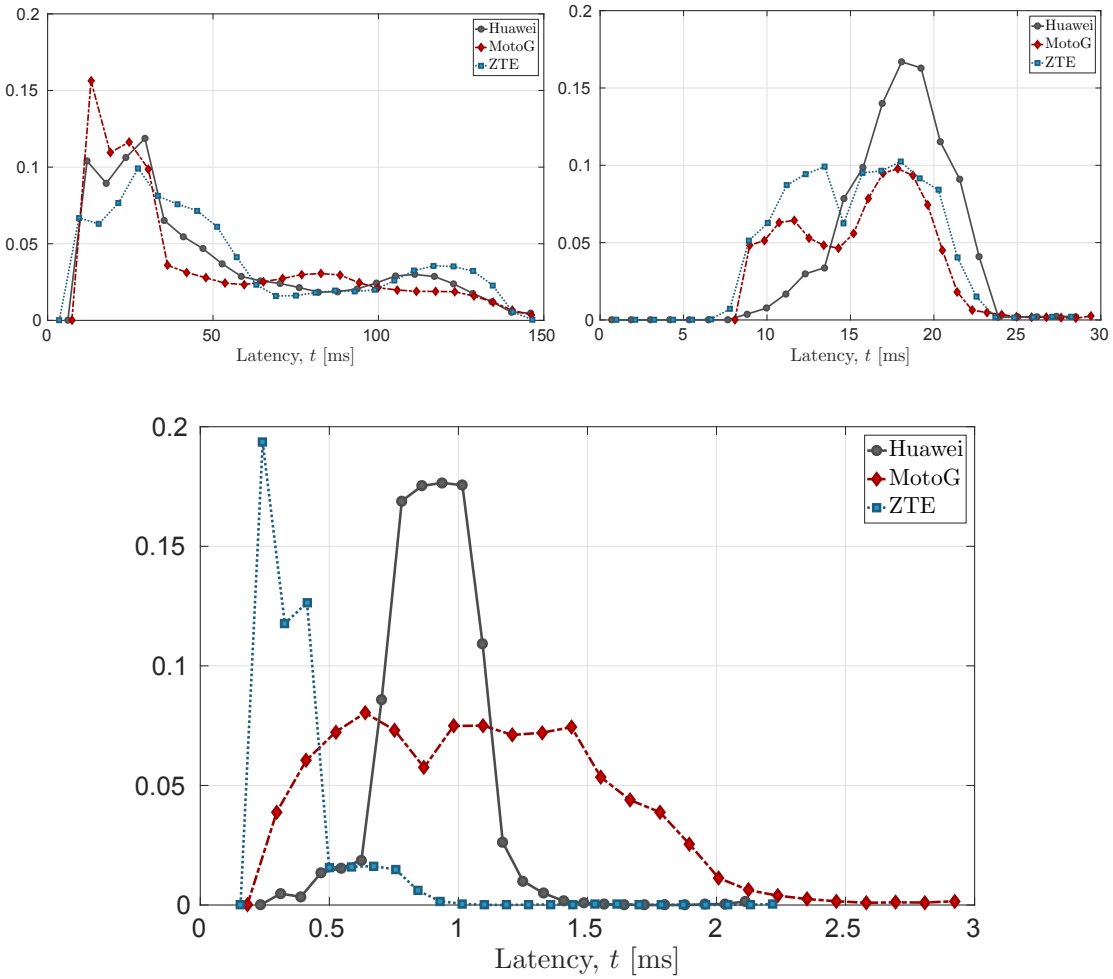
4.2. Experiment and Results

Both experimental campaigns follow the same base principle. We want to measure how much time after an event is registered in the physical layer, it is registered in the upper layers. It is important to note that systematic (*i.e.*, fixed) timestamping errors are not affecting the accuracy of the estimators we will present in the following chapters. For example, we suppose that there is a delay of exactly X ms between a packet arriving at the PHY and being registered by tcpdump at the IP layer. If we receive a group of packets, each one of them will be registered X ms later to the IP layer. Our bandwidth estimators, as we will elaborate in the sequel, require the time difference between the arrival of the first and the last packet of the group. This time difference is unaffected by the fixed delay.

On the other hand, the variability of the delay between the layers (*i.e.*, jitter), has a detrimental impact on our bandwidth estimators. Thus, in the experiments of this section, it is not needed to have perfect synchronization between the participating devices, since we want to track the variability of delay. We time the packets in the following way. In each measurement, we identify the packets related to the transmission at every layer. The first packet of the transmission at every layer, sets the beginning of time for this layer. In other words, we consider that the first packet arrives at time 0 ms. Therefore, we are able to track jitter between the layers, without perfect synchronization. Each experiment lasts a few seconds at most, consequently clock drift is ignored in this study.

4.2.1. Measuring Layer Latency (Isolated Transmission Test)

First we try to identify the impact of each layer in the perceived latency of reporting the arrival of a single packet. We send a series of 500 byte packets with a periodicity of 400 ms. The parameters were chosen in order to easily isolate our packets from competing traffic and LTE control messages, as well as distinguish the original transmission of a packet from potential retransmis-



(c) Phone latency (t_P)

Figure 4.3: Empirical probability density functions of the latencies observed in the downlink (figure from [1]).

sions. Further, the small packet size allows the encapsulation of data in a single "transmit unit" at each layer.

Figure 4.2 presents graphically the experimental process in the downlink scenario for a single packet. Each arrow represents the event timeline of a specific layer. The filled marker represents the timing of a TCP data packet and the empty marker the timing of its acknowledgement. We keep track of the time difference between the data and the ack and refer to this data-to-ack time as "latency". t_S, t_C and t_P is the latency at the IP layer of the server, the LTE PHY and the IP layer of the phone respectively. We cannot define latency at the application layer. Instead, we measure the time difference between the arrival of the IP packet and data appearing at the application and we denote it by t_A.

Figure 4.3 presents the Empirical Probability Density Function (EPDF) of t_S, t_C and t_P.

The latency measured at the server (Figure 4.3a) is the sum of:

- The delays caused by two Internet traversals.
- Two LTE scheduling delays (downlink first and then uplink).
- Phone processing (chipset time plus protocol stack traversal in the kernel).

The latency at the cell downlink (Figure 4.3b) starts when the downlink LTE transmission is already scheduled and, as a consequence, it only contains:

- The phone processing.
- LTE uplink scheduling delays.

The latency at the phone downlink (Figure 4.3c) starts when the kernel receives the reception interrupt from the chipset and finishes at the ACK transmission to the communication interface.

We observe that the distributions of t_S and t_C are similar for all the phones. Therefore, we may exclude the impact of network traversals and LTE scheduling from the rest of the analysis and focus solely on t_P. This is also an indication that they do not impact the data rates estimates we will present in the next Section.

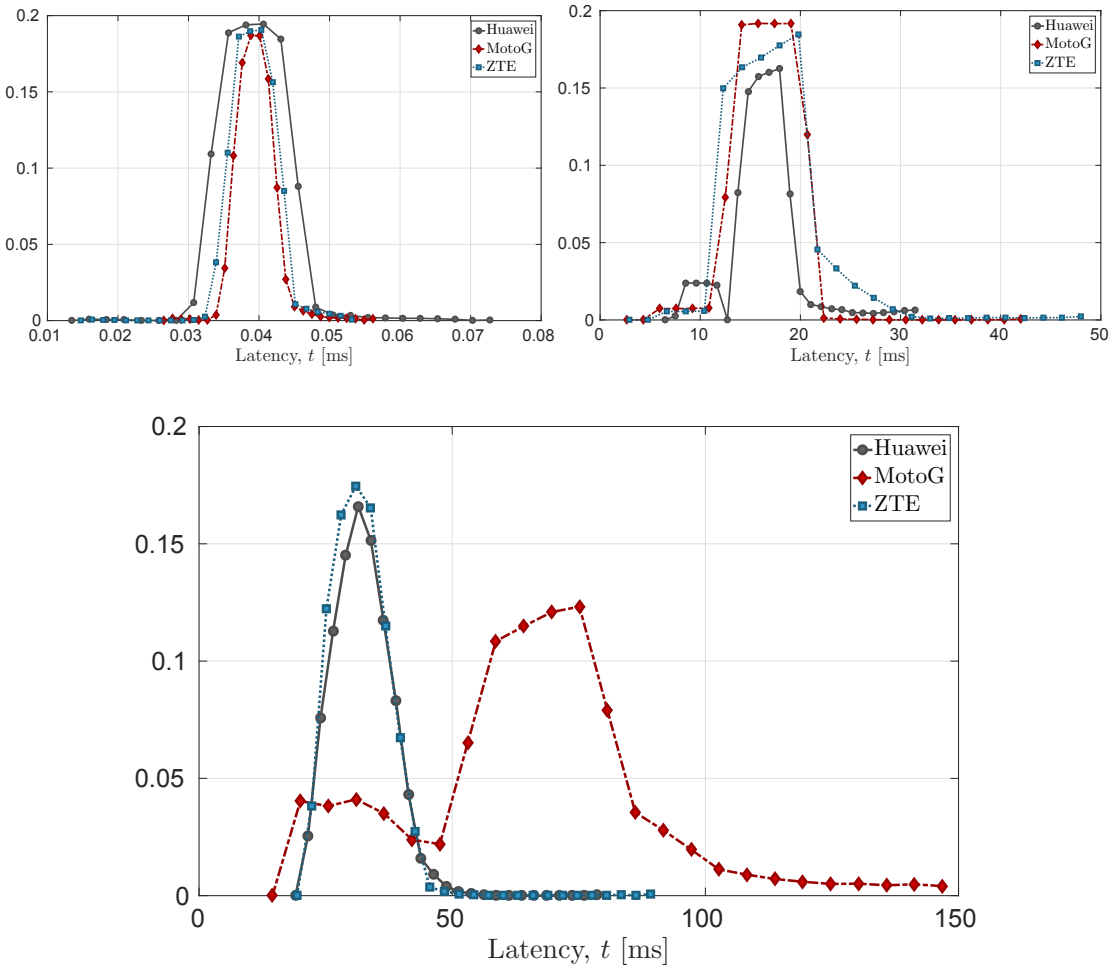
The main remarks from Figure 4.3c per phone are:

- **MotoG** exhibits a wider distribution of its latencies ranging from 0.3 to almost 2 ms.
- **ZTE** exhibits a low latency variability and a single peak around 0.5 ms.
- **Huawei** exhibits latency closer to the length of the LTE TTI around 0.9 ms. This may cause data rate overestimation.

In general, chipsets with short and deterministic latency achieve more accurate and precise data rate estimation, as we will present in Section 4.2.2. A higher latency makes it hard to discriminate the arrival at the phone of distinct LTE transmissions. Consequently the higher layers observe what appears to be smoothed arrivals and therefore provide less accurate bandwidth estimators.

We repeat the same experiment in the uplink scenario, where the phone is transmitting traffic to the server and receives the related ACK. In this scenario, the latency measured at the server is just the server processing. It should be noted that the kernel delay at the server is a mere 50 μs, with very low variability. Thus, it is reasonable to expect the t_P of the downlink scenario to improve in the future, as phone hardware becomes better, in effect allowing for better data rate estimators. At the cell vantage point, the latency includes the server processing delay and the two Internet traversals. Finally, the latency at the phone includes both uplink and downlink LTE scheduling, two Internet traversals and the server processing.

Server and cell latencies can be excluded from our analysis again, since as can be observed in Figures 4.4a and 4.4b, all phones exhibit identical latencies at these vantage points. We may attribute most of the variability observed in Figure 4.4c to the LTE uplink scheduling delay, which



(c) Phone latency (t_P)

Figure 4.4: Empirical probability density functions of the latencies observed in the uplink (figure from [1]).

in general is higher than the donwlink scheduling delay. The expected uplink delay in Connected state is 20 ms. Given the periodicity of the packets though, when the test devices initiate the transmission, they are in DRX mode. As a consequence, the transmission suffers an additional delay while the interface sleeps. The exact DRX parameters vary among the devices, but we can observe that motoG seems to have the most conservative setup with a peak at 85 ms and a lot of variability. The other two phones, have a latency around 50 ms. Due to the wider distribution of latencies, uplink data rate estimators are less precise than downlink estimates by 10%.

4.2.2. Measuring Link Rate Estimation (Burst Transmission Test)

In this section, we will attempt to estimate the link rate at the various vantage points and then evaluate the accuracy and precision of the higher layer estimations compared to the ground truth of the physical layer. Each experiment consist of transmitting a series of data bursts at periodic

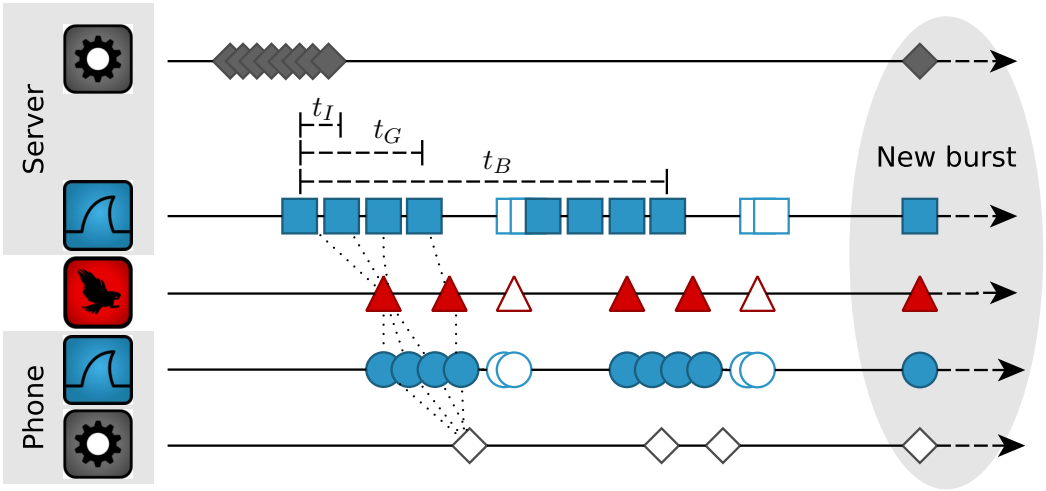


Figure 4.5: Communication diagram for downlink burst transmissions (figure from [1]).

intervals. The burst size is 100 KB for the downlink experiments and 30 KB for the uplink ones. These sizes ensure that, at the LTE MAC layer, the burst will be split among at least ten TBs, even at the most favorable conditions. We push the data burst at the socket in a single call. The data are then split among TCP packets before transmission. The server transmits them back-to-back in what looks like a packet train.

The packets get spaced even while they are at the server by TCP dynamics and processing delays. While they travel the Internet, they get spaced even more until they reach the antenna where they are stored at its buffer. Then a group of packets is transmitted in single TB to the target device. This packet group arrives at the device at exactly the same time. While the packets are processed by the higher layers one by one, they are spaced a bit. Eventually they reach the socket buffer, where their payload is stored until it is delivered to the application.

Figure 4.5 presents a diagram of this process. We are interested in identifying at higher layers the packets transmitted in the same TB. To this end we define the following parameters. The interarrival time between two consecutive packets at the same layer is t_I. The time difference between the fist and the last packet of a burst is t_B. We define "packet group", as a sequence of packets that we believe were transmitted in the same TB. In practice, we identify groups as packet sequences which have interarrival times less than or equal to τ. τ is bounded by the LTE TTI of 1 ms, therefore all the t_I of a group have the property: $t_I \leq \tau \leq 1$ms. τ is unique to each phone and depends on how much the different layers contribute to the latency. We plot the CDFs of the interarrival delays for the higher layers and choose the value of τ depending on where the plots do "knees" and flat regions[1].

The concept of "packet groups" is more clear in Figure 4.6, where we plot the arrival time of packets against the cumulative size of received data as captured by tcpdump. Packets that arrive

[1] We do not present these plots in this thesis.

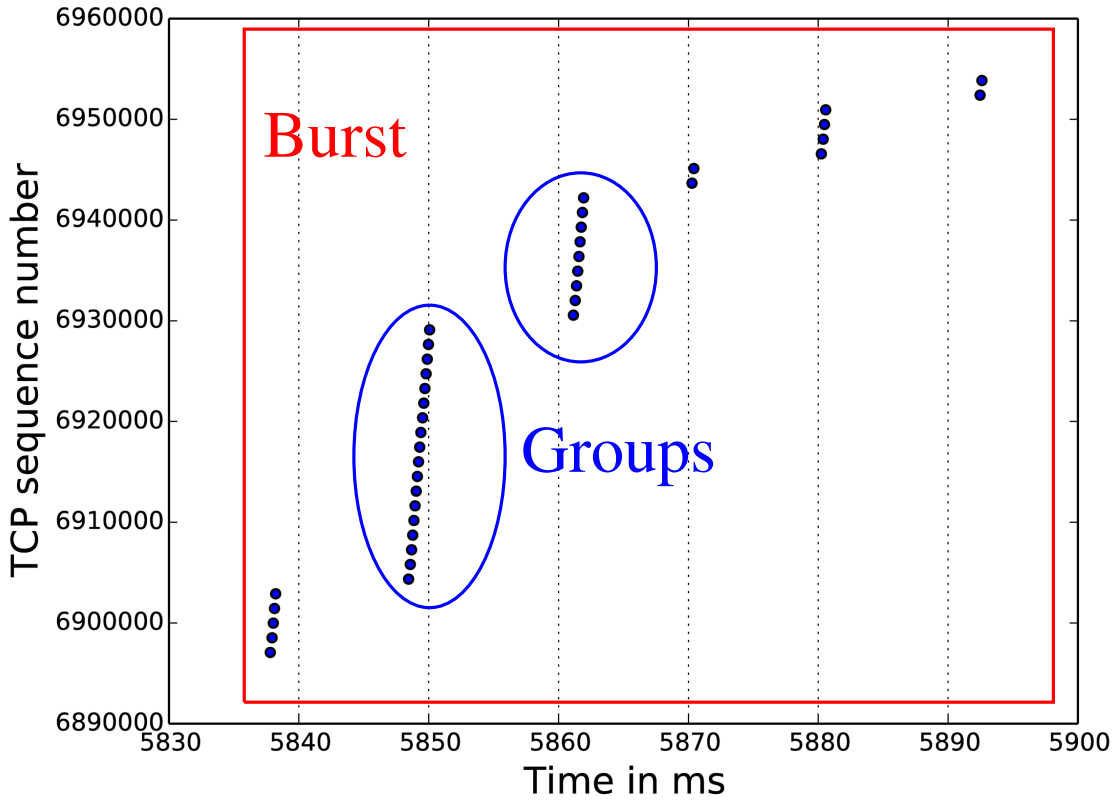
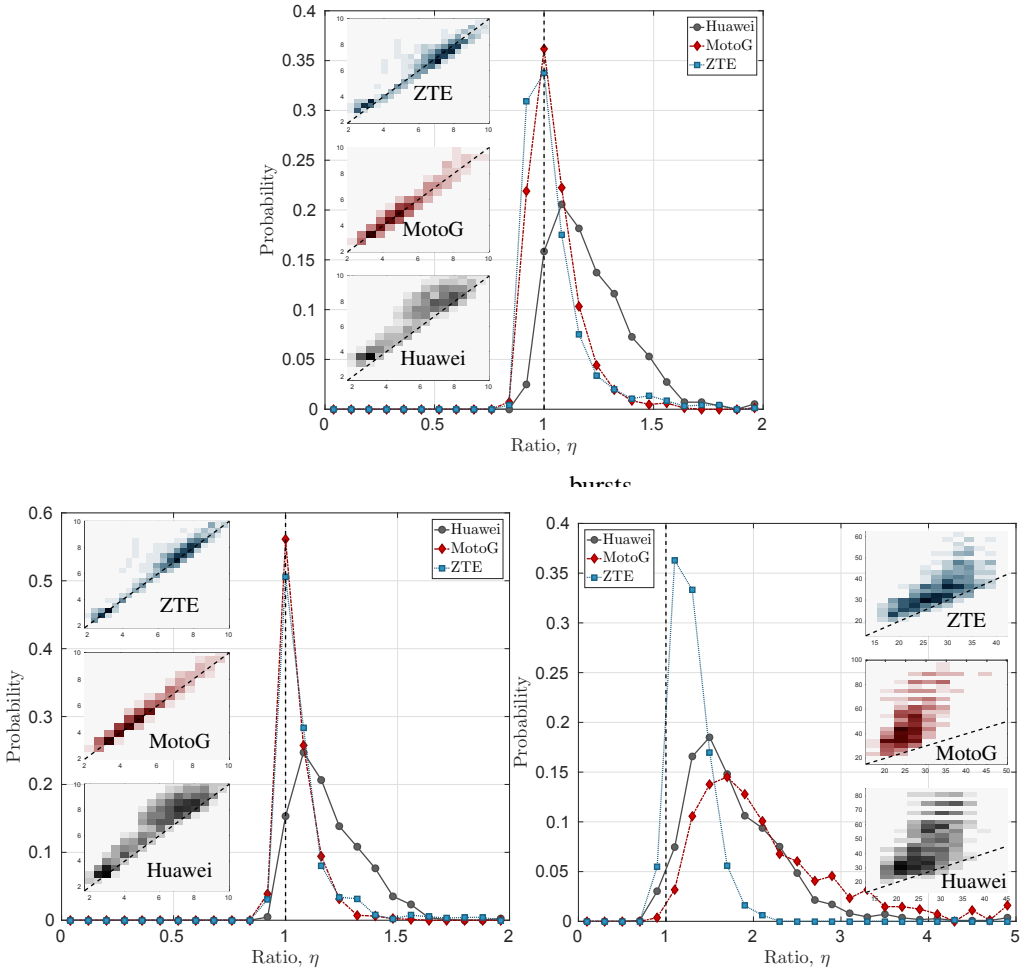


Figure 4.6: Inside a burst of packets, we may identify Groups that arrived at the phone simultaneously, encapsulated in the same TB.

in the same TB are close together forming a group, separated by several ms from the next group of packets transmitted in a single TB. At the LTE PHY layer, a group is a sequence of one or more consecutive TB transmissions as reported by the scheduling information. The time difference between the first and the last packet of the group is symbolized as t_G in Figure 4.5. The empty markers of Figure 4.5 stand for the ACKs of data packets.

The data rate is calculated as the fraction of data size and time duration. At every layer we may calculate the data rate on either groups or bursts. In the sequel we compare the data rate measured by the LTE sniffer, r_0, with the data rate measured by Application–bursts, Kernel–bursts and Application–groups. The results are summarized in Figure 4.7, which presents the empirical PDF of the ratio $\eta = r/r_0$, where r is the higher layer data rate estimation. Ideally, we would want values as close to 1 as possible. The smaller figures show the density of estimators in a system where the x-axis is the ground truth and the y-axis is the higher layer estimate. Ideally, all estimators should fall on the $y = x$ line.

The variability of the results proves again that different phones have slightly different biases. Among others, we examine the accuracy and precision of the estimators. Accuracy reveals systematic errors. The measure of accuracy in Figure 4.7 is the distance of the EPDF peak from 1. Precision quantifies the random errors and it is represented by how wide are the distributions.



(b) Kernel–bursts (c) Application–groups

Figure 4.7: Comparison of the estimator ratios computed on burst by the application (a) and the kernel (b) and on groups (c). The small plots on the left show estimator densities: the x-axis is the cell ground truth and the y-axis the estimate (figure from [1]).

Comparing Figures 4.7a and 4.7b, we observe that kernel measurements are slightly more precise. This is to be expected since, they are performed at a vantage point closer to the physical layer. Despite that, even applications are able to provide good enough estimators, with Huawei, the worst performer being able to achieve 85% accuracy and 82% precision.

Each phone shows a unique bias and all have a tendency to report slightly higher bandwidth values. Consequently, if a measurement campaign requires high accuracy, it is advised to calibrate the phones used. MotoG and ZTE achieve higher accuracy and precision as is expected, based on their performance at the latency test of Section 4.2.1 We further observe that accuracy and precision are independent of the actual data rates. The slightly larger width in the measurement point distribution at higher rates is expected since the same percentage error causes a larger absolute error at higher rates.

We observe that group based estimators have a tendency to severely overestimate data rate.

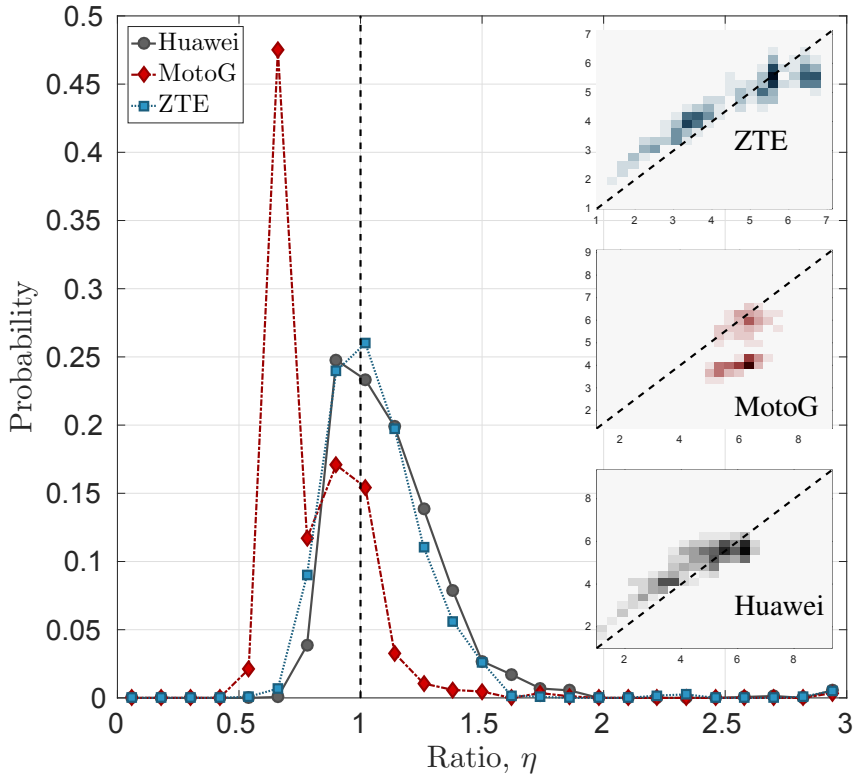


Figure 4.8: Estimator ratios computed on burst in the uplink, as measured in the kenrel level (figure from [1]).

In the sequel, we study group based estimators in depth and reveal the mechanics that cause this behavior. A brief explanation is that the groups have usually very small t_G, which combined with the inability of the phone hardware to provide accurate packet timestamps and the ignorance of group spacing times make it impossible to generate an accurate estimation. In scenarios where we try to opportunistically generate traffic estimators, by passively sniffing traffic created by third-party applications, usually only groups are available. To this end, Chapters 6 and 7, propose techniques that may use group related information to provide trustworthy results. On the other hand, the estimator values provided by bursts are averaged over a longer period of time, thus are less affected by artifacts and inaccurate timestamps. As a consequence, they may provide more accurate and precise estimations. In the rest of the analysis, we focus on burst based estimators.

We repeat the same experiment in the uplink and report the results of burst based estimators observed at the kernel level in Figure 4.8. As expected, by the analysis of the previous section, MotoG is the worst performing device, underestimating the real speed by 30%. On the other hand, the other two devices achieve accuracy of at least 93%. We observe though that the PDFs are wider than the downlink case, resulting in reduced precision of about 70% for ZTE and Huawei and 60% for MotoG. Consequently, we reach again the to the conclusion that if low margin of error is required, the measurement devices should be calibrated first.

4.3. Summary

This is the first study exploring the accuracy and precision of bandwidth measurements performed directly on a smartphone. In most cases, the value reported based on observations at the higher layers is close to the value observed at the physical layer, but if a low margin of error is needed, then the phones used should be calibrated, since hardware differences have a measurable effect. In general, downlink measurements have a smaller error compared to uplink. Smaller error is also observed when the traffic is monitored at the kernel level, instead of the application level and when bandwidth samples are generated based on bursts of packets rather than groups.

Chapter 5

Challenges in Performing Low Layer Mobile Measurements

This chapter analyses in detail how the mechanisms described in Sections 2.3 and 2.4 affect our attempts to accurately measure traffic and serves as the foundation on which Chapters 6 and 7 build upon. We also justify why the packet pair approaches mentioned in Section 3.1.2 are not suitable for use on mobile phones. The tools we will present are based exclusively on phone side measurements, and this imposes some limitations. Our main objective is to have a view of packet arrivals at the physical layer, but we can only know what tcpdump is reporting at the IP layer. Thus, we explore the measurement artifacts that distort the IP layer view of the traffic.

5.1. Measurement Artifacts

Figures 5.1 to 5.6 present time-sequence graphs of packet arrivals to a smartphone, as they were captured by the traffic sniffing tool tcpdump. The time values represent time since the first packet of the download arrived and when the related packets were captured by tcpdump. First, we examine what the arrival patterns of IP packets look like under ideal conditions. Figure 5.1 shows the arrival of packets to an LTE smartphone, as captured by tcpdump. In this experiment, we are saturating the link and observe its behavior during TCP steady state. Note that the TTI of LTE is fixed to 1 millisecond. It is easily observable that the packets arrive in groups that have about the same duration as the TTI. Between these groups of packets, the smartphone is not allocated resources, thus nothing is received. The size and temporal spacing of the groups depend on the channel quality of the UE and the congestion level at the base station.

In our traces though, we frequently observed measurement artifacts that are unrelated to the scheduler and are due to the following reasons.

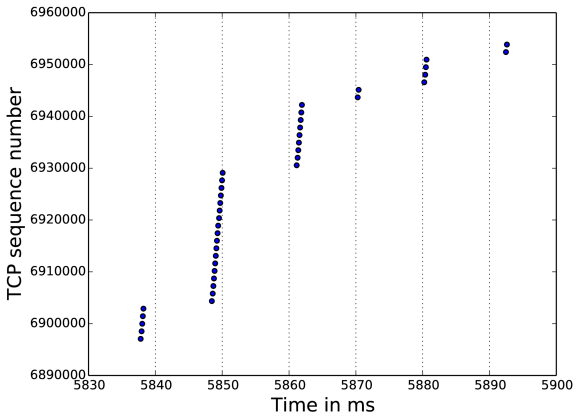


Figure 5.1: Link saturation traffic over LTE during the steady state of a TCP flow.

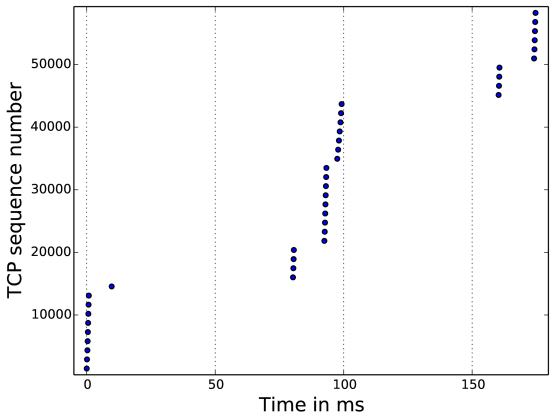


Figure 5.2: Arrival of the first packets of a TCP flow over LTE.

5.1.1. Small Congestion Window Values During the Slow Start

The perhaps most well known artifact which our algorithms must handle is the grouping of packets during the slow start phase of a TCP connection. The servers that transmit data over TCP send bursts of packets to the client and wait for the related acknowledgments before sending more. This behavior is very prominent during the slow start phase of the transmission when the congestion window has small values. The gap in the transmission at the server side may cause an analogous gap in the transmission at the base station. During this time, the base station is not sending data to the recipient UE, because there are not data in the dedicated buffer. This is visible in Figure 5.2, which illustrates the delivery of the first packets of a TCP flow over LTE. In two occasions, consecutive TBs are received with a delay in the order of tens of ms. We also observe in this example, that the total number of packets delivered in the groups that arrive at about 75 ms is bigger than the number of packets in the first set of groups (the second group has just one packet) at 0 ms. This is caused by the exponential growth of the congestion window. Eventually, the congestion window is large enough that the we observe a continuous stream of incoming packets and this effect diminishes. The specific flavor of TCP is not important, since all of them use initial congestion window values that create this effect. The Round Trip Time (RTT) is larger in 3G networks, compared to LTE, so, the impact of this TCP behavior is slightly more pronounced.

5.1.2. Infrequent Polling for Incoming Packets

IP packets arrive at the UE as part of a TB alongside other IP packets. An ideal method to measure the downlink speed then would require registering the exact size and timestamp of each TB. The related information is only available at the eNodeB, as we have explained in Chapter 2, so this is unfeasible. The lowest level from which we can extract network information is the kernel, where we register the time and size of all the IP packets. Usually packets are registered

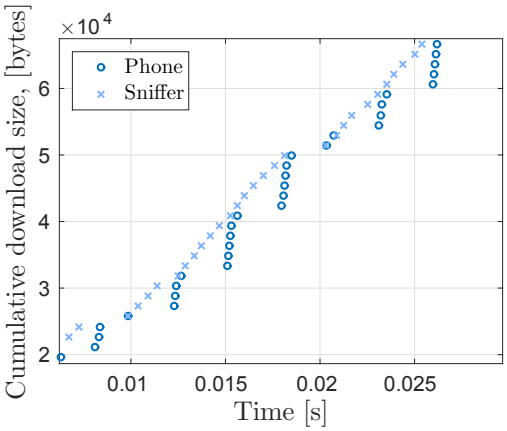


Figure 5.3: WiFi experiment of a phone with infrequent polling of the NIC.

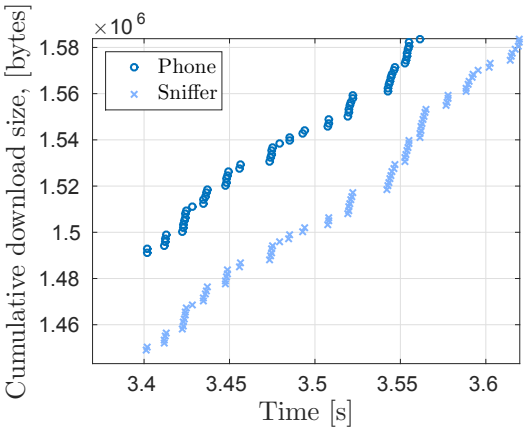


Figure 5.4: WiFi experiment of a phone unaffected by polling.

at the kernel with a noticeable delay, compared to their arrival at the NIC. In general, tcpdump registers packets as soon as they arrive in the kernel, but in case of high network load, the kernel might choose to delay polling in order to reduce packet processing overhead.

We have conducted a small scale experiment to assess the effect of polling on a variety of phones, when both the WiFi and the LTE interface are used. When the LTE interface is active, packets are reported in groups similar to the ones visible in Figure 5.1, in all of the phones. The pattern is always similar with some minor variations on the size and spacing of the groups, depending on how powerful the hardware is. For the WiFi experiment we use the 802.11g protocol without packet coalescing, to ensure that each MAC frame encapsulates exactly one IP packet and there is no grouped transmission of packets. We also set up a WiFi sniffer, which provides more accurate timestamps to monitor the exchanged traffic and provide the ground truth. In Figures 5.3 and 5.4, we show WiFi traces captured by the sniffer and the phones during high speed downloads. We observe that different phones may exhibit a very different behavior in the WiFi case. The WiFi sniffer always reports a continuous delivery of packets "in the air". Some phones report the packets in the same grouped fashion as the LTE experiments, whereas others report continuous delivery of packets. Based on these observations, we conclude that the pattern of packet arrival on WiFi seems to be greatly dependent on the phone specifications. The arrival pattern in the LTE case is determined by the grouped delivery of packets in the physical layer, but the timestamping accuracy of each packet is related to the phone hardware. More powerful phones are less affected by the polling problem, but even in this case, the delay shows slight variations. Since this delay is very small, it is not significantly affecting our techniques, whose adaptive and statistical nature tries to countermeasure it.

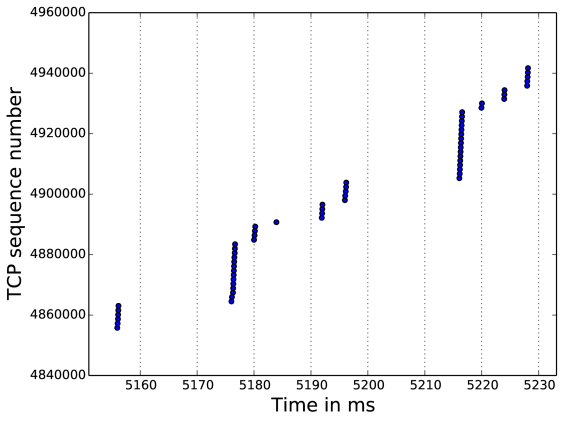


Figure 5.5: Some packets may be registered with a noticeable delay. Experiment over 3G.

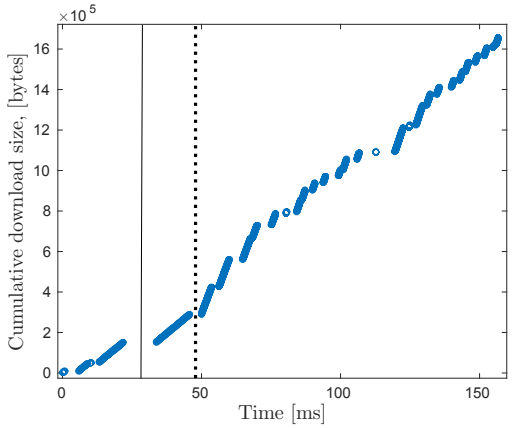


Figure 5.6: Arrival of high speed UDP CBR traffic over LTE.

5.1.3. Weak or Busy Phone Hardware

It is quite common for packets to be delivered to the phone but not delivered to the higher layers until several milliseconds later, alongside all the other packets that have been received in the meantime. This is usually observed in cases of high capacity and/or high CPU utilization. This behavior is very evident in Figure 5.5, which depicts the TCP steady state of a 3G download, at a speed close to the theoretical maximum the measurement device can support, without any other process stressing the CPU. Similar behavior is observed over LTE. According to the server side trace of this download, the server transmitted all the packets that are visible in the figure almost "back-to-back". Also, the phone trace showed a steady rate in the delivery of packets. But at times 5175 and 5215 ms we observe a gap in the delivery of packets and then the delivery of an impossibly large group. Packets were actually delivered during these gaps, but were registered all together when the CPU was able to process them. When the CPU is busy, this effect can be observed at lower speeds.

5.1.4. Slower Speed During the First Packets of a Flow

We have noticed that when a UE may achieve very high speed, there is a significant difference in the arrival rate of the first few hundred packets of a flow and the arrival rate of the rest of that flow's packets. The difference is present even if we take into account the reduced rate of the slow start phase of TCP, in case the flow is TCP. We have observed this phenomenon in traces gathered in the networks we used to evaluate our tools, as well as other European mobile networks and across different devices. In order to get more insight, we have done a small experiment in a Spanish LTE network, where we send CBR UDP traffic and monitor the arrival rate as reported by the mobile. When the server transmits traffic at a rate smaller than 25 Mbps, there is no difference in the arrival rate at different parts of the flow. If the rate of the server is higher than 25 Mbps, the first part of the flow (usually the first 150 to 300 packets) has an arrival rate 25%

to 50% lower compared to other parts of the same flow. A characteristic example is presented in Figure 5.6, where the speedup is easily observed if we study the angle of the groups. The angle, which represents speed, is lower on the groups on the left side of the dashed line compared to the groups on the right side of the dashed line. More specifically, the arrival rate of the packets located on the left side of the continuous line [1] (first 178 packets) is almost half the rate of the rest of the packets on the right side of the continuous line.

If the transmission pauses for a few tens of ms, the same effect is observed upon restart. Even though we do not present a dedicated experiment for a 3G network, our traces indicate that this phenomenon is even more prominent in 3G. An independent team of researchers [60], who conducted measurements in the same German network we used to collect our traces, observed that the first packets of a flow experience a considerably higher delay compared to the rest, when the rate at the server is higher than 20 Mbps. This effect causes reduced speed during the first part of the flow. At the time of performing these experiments, we were unable to investigate this phenomenon further, due to the lack of any means to monitor physical layer or mobile network specific information. We believe that it can be attributed to an operator configuration.

5.2. Packet Pair Issue

The previous characteristics of mobile networks and phone hardware make the use of traditional packet pair techniques infeasible. Any two packets that would make a packet pair are in either of the following cases.

Transmitted in the same TB. In this case the packets arrive more or less at the same time to the UE, since all the information included in the TB is transmitted in parallel using multiple carrier frequencies. The lower protocol layers of the UE ensure that they are delivered to the higher layers in the right order, while also assigning them slightly different timestamps. Consequently, sniffing tools like tcpdump perceive them as arriving with a tiny time difference, in the order of a few hundreds of microseconds. A capacity estimation based on these packet pairs would greatly over-estimate the real value of the capacity.

Transmitted in different TBs. In this case, the packet pair consists of the last packet of a TB and the first packet of the following TB. Thus, the capacity value is greatly underestimated, since the measured dispersion is the dispersion between the TBs and each TB is very likely to be able to encapsulate more than one IP packet, which is not reflected in the measurement. If there is exactly one packet per TB, then an accurate estimation is possible, but we observed that in the majority of the cases each TB contains multiple packets.

[1] In order to avoid the including the visible "gap", which is a measurement artifact, in the calculation, we calculate the speed for the first packets of the connection based on the arrival of the packets before the "gap".

5.3. Packet Trains Issue

Packet trains are also problematic. They cannot be used in a passive scenario because the server transmits packets on the receipt of ACKs and the application requirements, so the trains will have variable length. The number of packets in each TB may be different, which results in similar problems to the ones described in the "packet pair" scenario. On some occasions all the packets will be transferred in the same TB and on others in multiple TBs.

It is clear that long-established packet dispersion techniques that were developed to detect the bottleneck link capacity in wired networks are not suitable for mobile networks, especially in regards to detecting the per user capacity. In the sequel, we will present the necessary modifications to this approach for it to provide reliable capacity estimations in mobile scenarios.

Chapter 6

Passive Mobile Bandwidth Classification Using Short Lived TCP Connections

The current state of the art solutions for bandwidth estimation require the transmission of link saturation traffic [15, 45, 46]. This practice is problematic considering the scarce resources of mobile devices. The focus of this study is the estimation of available bandwidth of mobile phones through passive traffic monitoring. We specifically focus on traffic generated during the initial few hundred milliseconds of the TCP "slow start" phase. At least 95% of the streams a mobile phone generates are TCP and the majority of them are so short lived that they do not exit the slow start phase of TCP [16]. In this chapter, we present a method that provides good bandwidth estimation under these challenging conditions. Further, our method is robust against measurement artifacts introduced by hardware-limited mobile devices, and the scheduling process of the base station. The run time complexity of our algorithm is $O(n)$ and the algorithm requires only the first packets of a TCP stream.

This information may be used to choose the most appropriate media quality in streaming scenarios, before the actual streaming commences. For example, a user may browse a library of available media before picking one to stream. We can take advantage of the minimal traffic the thumbmails generate in order to make an estimation of the bandwidth the user currently has. Thus, it is possible to pick the best media quality that has sufficiently low probability of causing buffering delays and slow startup.

The output of our algorithm can also be used as input to bandwidth optimization algorithms [7–9], to achieve more efficient usage of the mobile resources. Such algorithms are usually based on a resource prediction model that makes heavy use of time series information. We evaluate our algorithm by means of real-word experimentation, through LTE traces collected in 4 European countries (Germany [26], Sweden, Greece and Spain), using a variety of Android devices.

The rest of the chapter is structured as follows. Our algorithm and a comparison with similar tools are presented in Sections 6.1 and 6.2. Section 6.3 offers a discussion about the potential and the limitations of our solution. Finally, Section 6.4 concludes this chapter.

6.1. Algorithm

In this section, we introduce the bandwidth estimation algorithm. Our goal is to provide a tool that can estimate the available bandwidth of a mobile device by passively monitoring the traffic exchanged during the slow start phase of TCP, coping with all the artifacts described in Chapter 5. The core idea can be summarized as: we try to identify groups of packets that arrive at the eNodeB together and thus were transmitted to the UE at the maximum speed that the scheduler could allocate at that instance. For each such group we compute the bandwidth by dividing the group's total number of bytes by its time duration. Because this approach is very susceptible to artifacts, we apply a set of filtering techniques before a result is derived.

Ideally, we would want to extract traffic information from the NIC, but this is impossible without specialized drivers that vendors are very hesitant to release for public usage. Instead, we use rooted Android phones and the traffic sniffing program tcpdump [73] to record the time and the size of every IP packet reported by the kernel.

After we sniff the traffic and we organize it into flows based on IPs and ports. Packets related to TCP and TLS handshakes are ignored, since we are only interested in the data exchange part of the connection. On phones, most of the time only one TCP flow is actively downloading data [16], thus the chance of having overlapping flows is low. Even if there is an additional flow, it will most probably be in the slow start as well, or generate very low traffic. We may include these packets in the measurement, since our goal is to detect burst transmissions from the antenna.

An outline of the algorithm is presented in Algorithm 1 and its analytical presentation follows. For each incoming data packet P_i of a given flow, its size S_i and timestamp T_i are logged. The logging continues until a TCP FIN or RST packet is detected or until a few seconds have passed without any data exchange. The next step is to identify the groups of packets that were transmitted back-to-back from the server, which is done by locating unusually large delays between the arrivals of two consecutive packets. At this point, the timestamps of all the packets of the flow are available, so we derive their inter-arrival delays $D_i = T_{i+1} - T_i$. We ignore all the delays that are smaller than 1 ms, as such small delays imply that the packets were transmitted in the same TB and arrived at the same time at the NIC (i.e. packets that belong in the same "line", as shown in Figure 5.1). The remaining delays are either caused by the scheduler organizing IP packets into consecutive TBs (delay between the last packet of a TB and the first packet of the next TB) or the server having paused the transmission because the congestion window limit has been reached (delay between the last packet of a server group and the first packet of the next server group). We want to identify the latter.

Usually, a TB related delay is significantly smaller than a server group related one. However,

Algorithm 1: Algorithm outline.

Data: Array T of n timestamps of packet arrivals
Array S of sizes of n packets
Result: Estimation of the instantaneous available bandwidth class
$D, G, BW \leftarrow \emptyset$;
for $i = 1$ to $n - 1$
do
  if $T_{i+1} - T_i \geq 1$ ms **then**
    $D = D \cup \{T_{i+1} - T_i\}$;
$s \leftarrow 1$;
P_1 is the first packet of group G_1;
for $i = 1$ to $n - 1$
do
  if $T_{i+1} - T_i \geq \text{average}(D)$ **then**
    P_i last packet of group G_s;
    P_{i+1} first packet of group G_{s+1};
    $s \leftarrow s + 1$;
P_n is the last packet of group G_s;
for g in G
do
  if $T_{g_{last}} - T_{g_{first}} < 2$ ms **then**
    Delete g;
  else
    $BW = BW \cup \left\{ \frac{\sum S_g}{T_{g_{last}} - T_{g_{first}}} \right\}$;
return 75^{th} percentile of BW;

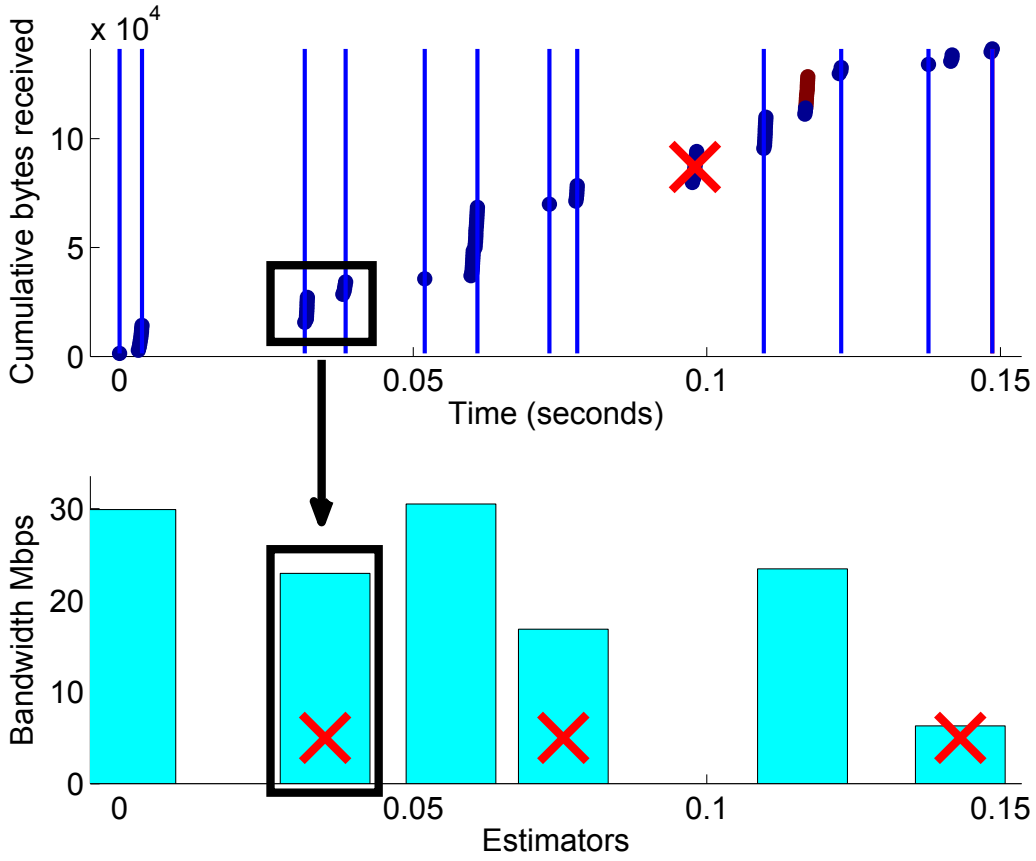


Figure 6.1: First 100 packets of a TCP flow. The identified groups are enclosed between two vertical lines and their derived bandwidth estimators are located right below them.

this observation applies only to the early stages of slow start. For larger flows, which reach a state of almost continuous arrival of packets the two kinds of delays are indistinguishable. Such flows are beyond the scope of our tool, since other established bandwidth estimation tools can be used in case a flow is that large. If a delay D_i is higher than the average delay, we assume that it is server related, thus packet P_i is the last packet of the server group G_s and P_{i+1} is the first packet of the next server group G_{s+1}.

The duration of a group is the time difference between its first and its last packet. Each group that has a duration of less than 2 ms is ignored. In order to generate a bandwidth estimator from a group, it should consist of multiple TBs. Groups with a total duration of less than 2 ms usually are single-TB groups. For such, the measured time duration is unreliable since the packets arrive at the same time at the physical layer. At the kernel level where they are reported, there is a small time difference, which, if used to compute an estimator, would yield very high values. Also, such groups can be indicative of artifacts from a weak CPU, where a lot of packets appear at once after a long period of inactivity. The upper part of Figure 6.1 presents the first 100 packets of a high speed download generated by the Speedtest application. Our algorithm was able to identify six

valid groups (marked between vertical lines), and one group that had to be ignored because of its small duration.

For each remaining group g a bandwidth estimation value is derived by summing its total number of bytes over its duration:

$$BW_g = \frac{\sum S_g}{T_{g_{last}} - T_{g_{first}}} \tag{6.1}$$

We then filter out the lowest 50% of these values. Such small values are usually caused by smaller groups that are unable to fully utilize the available bandwidth (usually the very first server group, which was generated with the smallest possible congestion window value). Other possible reasons are sets of packets that arrived a little later at the antenna because of cross-traffic, or the algorithm splitting groups too aggressively. In order to avoid the effect of very large outliers, the median of the remaining samples is the output of the algorithm. These outliers could be caused by ordinary polling delays and weak hardware artifacts, which may significantly alter the timestamps. Effectively, these last two steps are implemented by picking the 75^{th} percentile of the values. In the example of Figure 6.1, the estimator values of each group are calculated in the lower part. The values of the 2nd, 4th and last group are ignored as part of the lowest 50% and the final result is the median of the remaining ones: about 30 Mbps.

This algorithm is so lightweight that it can be used on any modern mobile hardware with minimal impact on its resources. Not only is its complexity $O(n)$, but also n is bounded by the number of packets, which is at most a few hundred. The algorithm is designed to provide results for flows that have a number of packets that ranges from many tens (about 60-80) to a few hundred.

6.2. Comparison with Bin-Based Tools

In this section we assess the accuracy of our algorithm. We collected traces from 4 European countries (Sweden, Germany, Spain, and Greece) using several FDD LTE devices. Our traces were collected by measuring traffic generated by automated tests and by volunteers who performed their usual tasks on our instrumented devices. In this way, we can test our algorithm in a variety of scenarios and configurations.

To the best of our knowledge, there are no other bandwidth estimation tools that can exploit the traffic exchange of the slow start phase. An alternative would be to compare the resources allocated by the eNodeB scheduler to the result of our algorithm. This can be achieved by accessing the logs of an eNodeB in order to fetch the exact timing of each TB transmission. Obtaining such information is very hard, since it requires access to network components that are regarded as commercial secrets by both equipment vendors and mobile operators[1]. Instead, we compare to a baseline bin-based algorithm similar to the one used in [45] and [15]. If the measurement was

[1] At the time of performing the experiments and analysis of this chapter, we did not have access to OWL

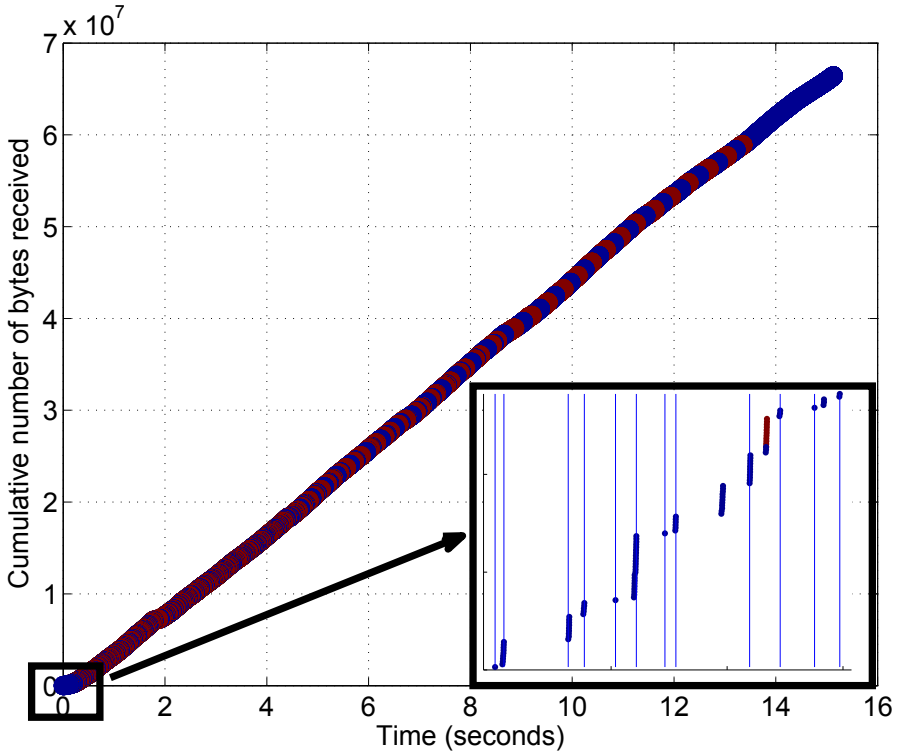


Figure 6.2: A trace generated by the Speedtest application while the UE was stationary and connected to an uncongested cell.

generated by the Speedtest application, we use the result of this application instead. The baseline algorithm works as follows: 1) the data exchange part of a flow is isolated, then 2) the flow is split into 100ms bins, 3) a bandwidth estimation for each bin is generated by dividing the data exchanged by its duration, 4) the highest 10% and the lowest 30% of the values, as well as bins with no data, are discarded in order to reduce the effect of slow start and measurement artifacts and finally 5) the average of the remaining bins is returned. When the baseline algorithm is used on Speedtest generated traces, the deviation from the Speedtest value is at most 8%. Therefore, we believe its results are close to those of Speedtest. We remark that our algorithm only has access to a very small part of the data, while the baseline algorithm has access to the full trace.

A comparison between our algorithm and an active measurement one should be done with caution, because they are designed to compute different metrics. The data exchange part of files in the range of 100 KB, which is the target of our tool, requires no more than 500ms even in slow connections and can often be completed in less than 200ms. On the other hand, an active measurement tool requires very big downloads, that must be active for 10 to 20 seconds, in order to provide trustworthy results. Thus, our tool measures instantaneous bandwidth and an active tool measures the average channel capacity over the duration of the measurement. The perceived bandwidth at the side of the end user may greatly vary over a 10 second period. This can be caused by the channel experiencing fast variations (e.g., moving car), changing cell congestion,

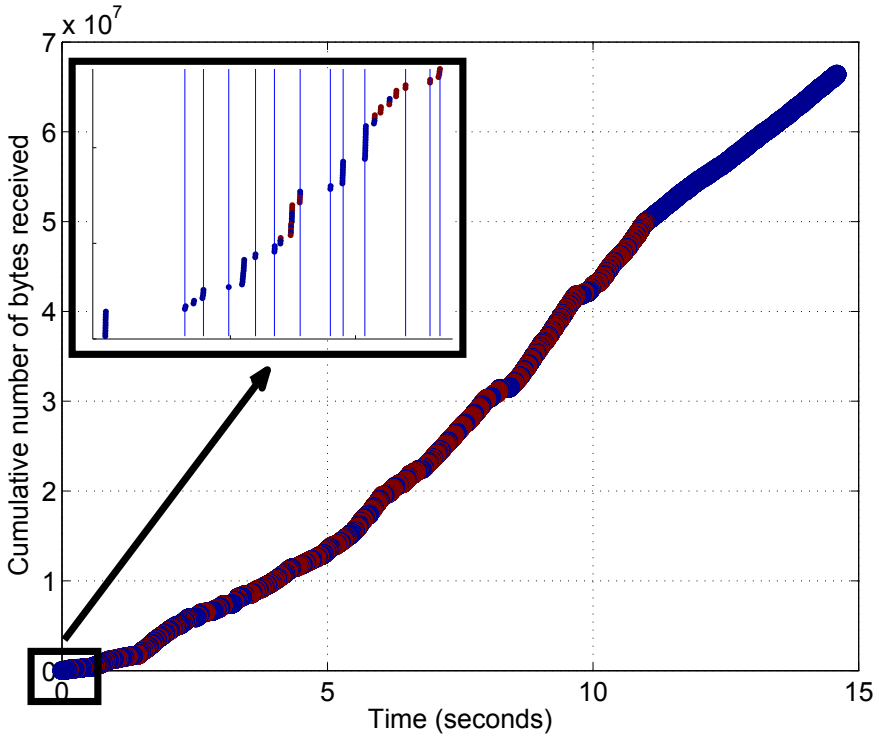


Figure 6.3: A trace generated by the Speedtest application while the UE was in car moving with 100 Km/h.

or packet loss. In such cases, it is expected that the results of the two tools differ. This effect can be seen in Figures 6.2 and 6.3, which present traces generated by the Speedtest application. Both traces generate a similar pattern in their slow start phase (presented in the the smaller figures), which is what our tool is designed to use as input. The trace of Figure 6.2 continues to have a stable packet arrival pattern throughout its duration, in contrast to the one of Figure 6.3. Consequently, the deviation of the resulting speeds between the two tools is several times higher in the second case compared to the first. In addition, it is reported that even well regarded tools that are meant to measure speed on the less dynamic "wired scenario" using a similar approach may have significant deviation between each other [74].

For the rest of this section, the result generated by our tool is based on the first 100 packets of a flow. We use this limit in order to represent what it would be able to track if the flow was a short-lived TCP download that finished before exiting the slow start phase. Figure 6.4 results are generated by a small subset of the collected measurements generated solely by the Speedtest application. Except for some cases that were caused by the phenomena described above, the deviation is within 45%. Our tool is consistently giving lower values, because of the effect described in Figure 5.6, since the speed achieved in most of the these traces, as reported by Speedtest, was mostly above 35Mbps.

Next, we filter our traces in order to keep flows that could be used by both our tool and the active bin-based estimator, described above. Thus, we reject flows that are too short, have too

few packets, are UDP, use network technologies other than LTE etc. Consequently, the size of our sample reduces greatly from a few thousand flows to a few hundred, since most flows are too short-lived for the baseline algorithm. This further highlights the importance of a passive tool that can be used when very low traffic is present. The deviation between the values of the two tools is presented in Figure 6.5. For the majority of the cases the deviation is less than 50%. The traces which exhibit a deviation of more than 100% are caused mostly by flows that exhibit traffic patterns not suitable for binning (e.g., video streams) or extreme cases of channel variation (e.g., usage on a train). For example, a video stream is unsuitable for binning, because it is very bursty—there are long pauses and short bursts of rebuffering. The binning algorithm samples using a fixed time interval of 100 ms. Thus in every burst the first and the last bin are mostly empty, resulting in significant underestimation. Building a more robust binning for the baseline algorithm is beyond the scope of this work.

For the purpose of selecting the appropriate bitrate for a streaming application, it is convenient to classify the resulting bandwidth into ranges. The number of classes and their limits are chosen with respect to bandwidth range categories that would make sense for a multimedia streaming application (bitrates of different stream qualities). Thus, the lower classes have significantly smaller range than the higher ones. To this end, we define five classes, that have the following Mbps ranges: 1) 0-5 Mbps 2) 5-10 Mbps 3) 10-20 Mbps 4) 20-60 Mbps and 5) higher than 60 Mbps. Table 6.1 presents how frequently our algorithm and the baseline match and by how many classes they differ, if they do not. We present both a comparison over all the traces and a comparison over only the traces that have traffic patterns more suitable for binning. Even though the two approaches have different objectives, in at least 60% of the cases they agree on the class. This is significant, considering that the result of our solution is derived with virtually no cost and there is no other tool that may provide such information.

Class difference	Same	1	2	3	4
Traces suitable for binning	70%	29.6%	0.4%	0%	0%
All traces	60.29%	30.88%	8.46%	0.37%	0%

Table 6.1: How frequently our algorithm and the baseline match.

Our approach is optimized for cellular scenarios but we have also performed some limited experimentation with WiFi. The challenges of a WiFi scenario are different from those of a cellular one. In brief, 1) the channel is half duplex, 2) usually a centralized scheduling entity is absent, 3) the significantly lower RTT compared to cellular networks may make the detection of burst transmissions by the server harder, 4) the variable timing of the back-off mechanism, 5) the potentially weak hardware and the great variation of polling behavior, as presented in Chapter 5 and finally 6) the presence of broadcast traffic.

Despite the above, the same version of the algorithm, applied to a small WiFi trace was able to agree with the bin-based benchmark on the bandwidth class in at least 50% of the cases. Also, the deviation between the values of the two tools was again less than 50% in the majority of the

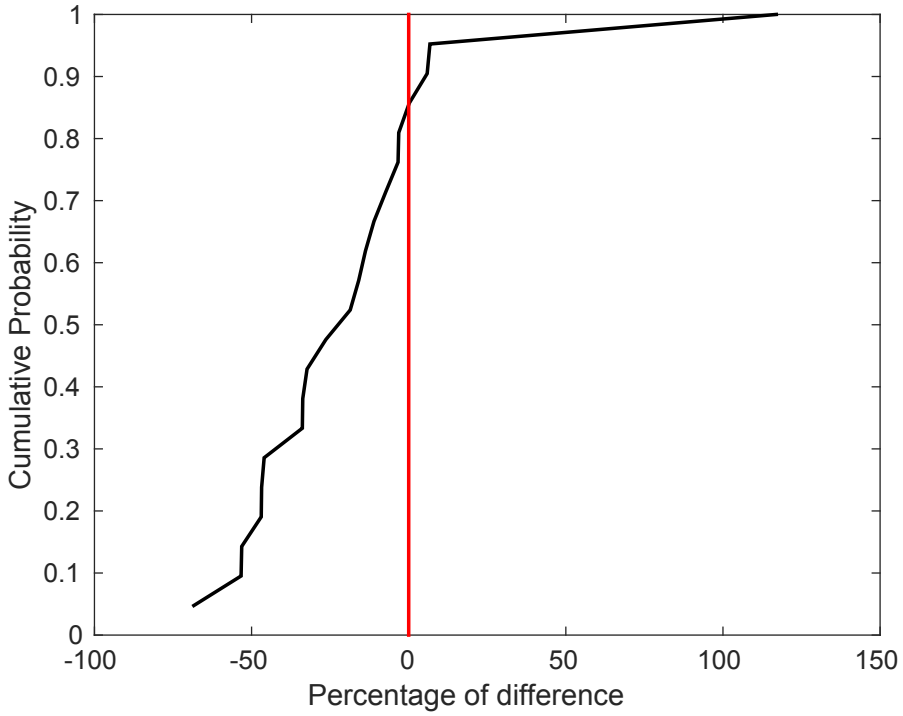


Figure 6.4: Percentage of difference between the instantaneous bandwidth measured by our tool and the average bandwidth measured by the Speedtest APP.

trials.

6.3. Discussion

This section highlights some important aspects of our solution. The small footprint of the algorithm makes it possible to run it as a background service without influencing the OS's resource utilization. For example, it could be integrated in the interrupt or polling functions triggered upon packet arrivals. As mentioned in [50], some of the services that generate the flows we monitor might be rate limited on the server side or by a bottleneck link other than the antenna. In this case the tool does not measure the available bandwidth that the eNodeB can allocate to the user, but rather the rate of the server or the bottleneck link of the path. Thus, as with other passive techniques, the output of our algorithm should be seen as a lower bound. Also, a big portion of flows do not generate enough traffic to be used for a trustworthy estimation even for our tool. This includes applications that by definition use use very low traffic (less than 10 packets per 100ms), like chat applications (WhatsApp), or notification services like "Google cloud service". Of course, if a user receives a media file in one of these apps, the traffic generated then is sufficient.

If the reported bandwidth is above 25Mbps, it is possible that the actual bandwidth that could be achieved by larger flows is significantly higher because of the phenomenon presented in Figure 5.6, where the first few hundred packets of a flow have an artificially slower speed than the

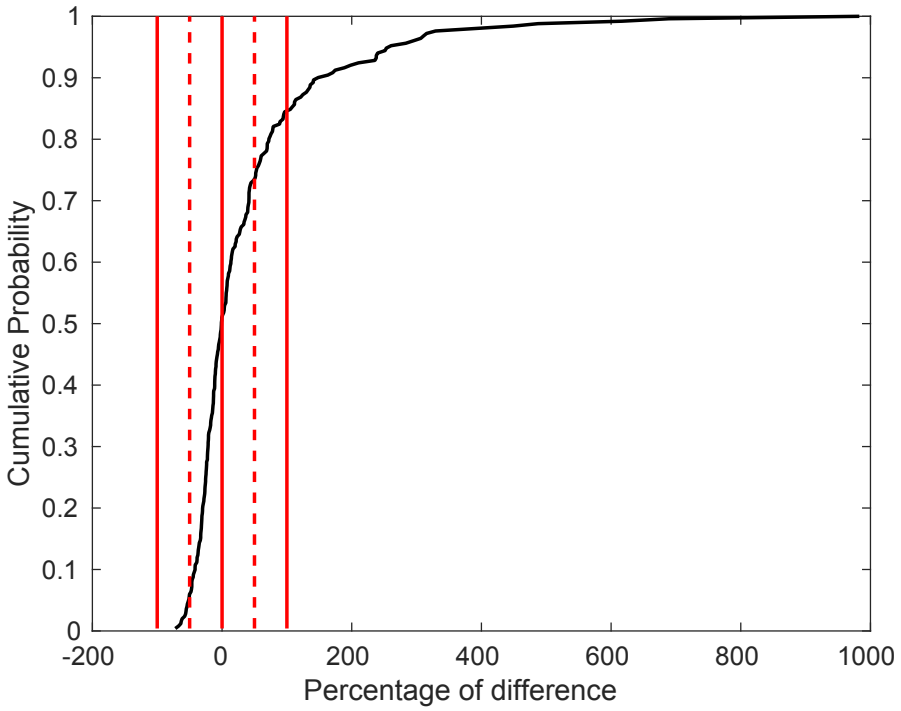


Figure 6.5: Percentage of difference between the instantaneous bandwidth measured by our tool and the average bandwidth measured by a bin-based estimator. The solid and the dashed lines mark the 100% and 50% limits respectively.

rest, which are able to achieve the full possible speed. Also, it is possible that the slow start phase of TCP cannot (even momentarily) saturate the link enough for the estimators to get such high values. Our tool is meant to be used to optimize the QoE of media streaming applications and not to provide a highly accurate bandwidth estimation. An underestimation of the true value at such a high bandwidth class is not going affect the media application. Even the most demanding video streaming applications, like the Ultra HD quality of Netflix require bandwidth in the range of 25 Mbps [75]. Thus, any value above that is guaranteed to ensure minimum start up delays and uninterrupted playback, while offering the best possible stream quality. If desirable, the exact value of bandwidth might be obtained by another solution after the playback has started. This tool is designed to operate during the slow start phase of a TCP connection. If the flow enters the steady state, our tool is under-performing, because it will try to find server related groups of packets, when the incoming packets form a continuous stream. In such cases, it is better to use another approach that is designed to work on large flows. Our tool is meant to be used as a complementary solution to the flows that cannot be used by the existing bandwidth estimation algorithms.

Since our tool only tracks the size and time of the incoming packets, there is no privacy violation. The IP and port pairs are only used to identify when a flow is active and can be discarded after the measurement. Finally, when a user has a lot of small downloads within a short period of time, such as browsing web sites, the resulting estimations may be able to reflect the channel

variation.

6.4. Summary

We have presented a lightweight mobile bandwidth estimation tool able to estimate the bandwidth class of a UE by monitoring small flows. It is designed to be robust against various measurement artifacts introduced by the phone hardware and the scheduling process of mobile networks and is ideal for enhancing the Quality of Experience (QoE) of streaming applications. It can provide an estimation of the bandwidth available to a device, by just monitoring flows that precede a media streaming request, enabling the optimal selection of content bitrate. This solution is meant to be used alongside traditional bandwidth estimation tools that require access to large flows, thus offering reliable estimation for a great range of traffic. It was also evaluated with traces collected in 4 European countries with a variety of devices.

Chapter 7

Lightweight Capacity Measurements For Mobile Networks

Mobile capacity measurement is a well investigated topic in the literature, but, to the best of our knowledge, no lightweight or passive technique allows mobiles to collect frequent measures of their capacity. To fill this gap, this chapter proposes a simple technique which is able to measure the fast variations of the per user capacity and, from those, the expected end-to-end throughput.

Our goal is to provide a lightweight measurement technique that evaluates passively or with minimum impact the per user capacity variations over time in a mobile environment. This enables filter based prediction techniques and, consequently, prediction based resource allocation optimization. Source code for the tool can be found in the repository of the EU project eCOUSIN[1].

In order to do so we adapt packet train dispersion techniques by applying an adaptive filtering mechanism, which we show is effective in removing the impact of outliers due to bursty arrival and jitter, which are very prevalent in mobile environments. We validate the effectiveness of the solution through extensive simulation and "real world" measurement campaigns: our technique can achieve an accurate throughput estimate with as few as 5 % of the packets needed by other solutions, while making an error smaller than 20 %.

The rest of the chapter is structured as follows. We present our measurement technique in Section 7.1, in Section 7.2 we provide a first evaluation of our technique based on simulations, and in Section 7.3 we describe how we collected "real world" data to validate it. The results are discussed in Section 7.4 and Section 7.5 concludes the chapter.

7.1. Mobile Capacity Estimation

In the literature, the term "link capacity" refers to the transmission rate of a link, "path capacity" is the minimum transmission rate among all the links of the path and finally "link available bandwidth" refers to the spare link capacity (capacity not used by other traffic) [48]. Instead, we

[1] `https://ecousin.cms.orange-labs.fr/sites/ecousin/files/lightmeasure.zip`

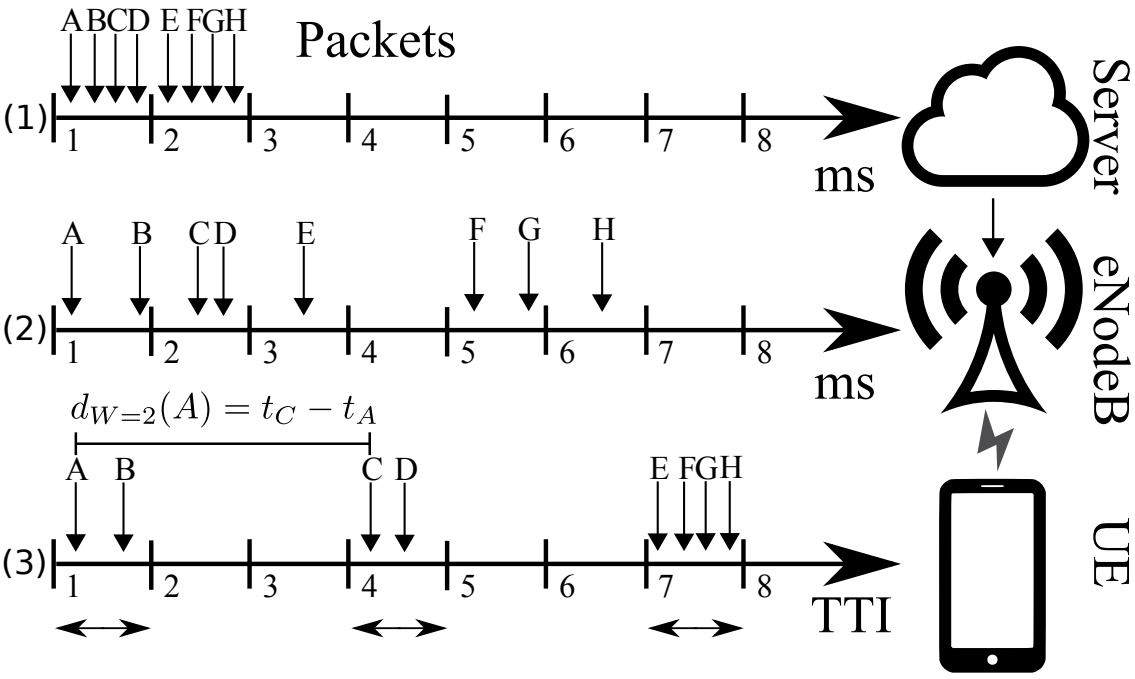


Figure 7.1: Dispersion of IP packets over the Internet. First, they are sent back-to-back from the server (1). After experiencing dispersion on the Internet, they arrive on the BS (eNodeB) (2). Finally, they are received in groups by the UE (3). The timelines (1-3) happen sequentially, one after the other, not in parallel. The horizontal arrows represent TBs allocated to the recipient UE.

are interested in estimating the maximum capacity that the scheduler of an eNodeB could allocate to a target user if he requested saturation traffic under a specific bearer. This metric is specific to cellular networks, we call it "per user capacity" and we symbolize it as C_U. For brevity, in the rest of the chapter we refer to it as "capacity". To the best of our knowledge, traffic flow templates are not used for generic browsing and multimedia traffic, which is the scope of this work. Thus, we can safely assume that all the measured traffic is using the default bearer, allowing us to ignore this variable. As we will analyse in the sequel, in practice, the measured C_U will often be less than the maximum capacity a user could be allocated. For this reason, the measured value represents the greatest lower bound of the user's capacity. We will show that this value is very close to the actual maximum, thus causing a slight underestimation of the true maximum per user capacity.

The wireless link is the last hop of a downlink path and the C_U of all the connected users is dependent on the cell congestion, the channel quality, the channel's bandwidth and the scheduling algorithm. It is usually the link of a path with the lowest capacity, that also contributes the most to the delay. On the other hand, the average end-to-end TCP throughput R, depends on the capacities and the cross traffic of all the links in the path, as well as possible rate adaptations at the server side, caused by the TCP mechanisms. The end-to-end TCP throughput is primarily determined by the link with the minimum spare link capacity, which in a mobile scenario is usually the RAN. We are interested in measuring C_U, since it is the metric that affects all the connections that the

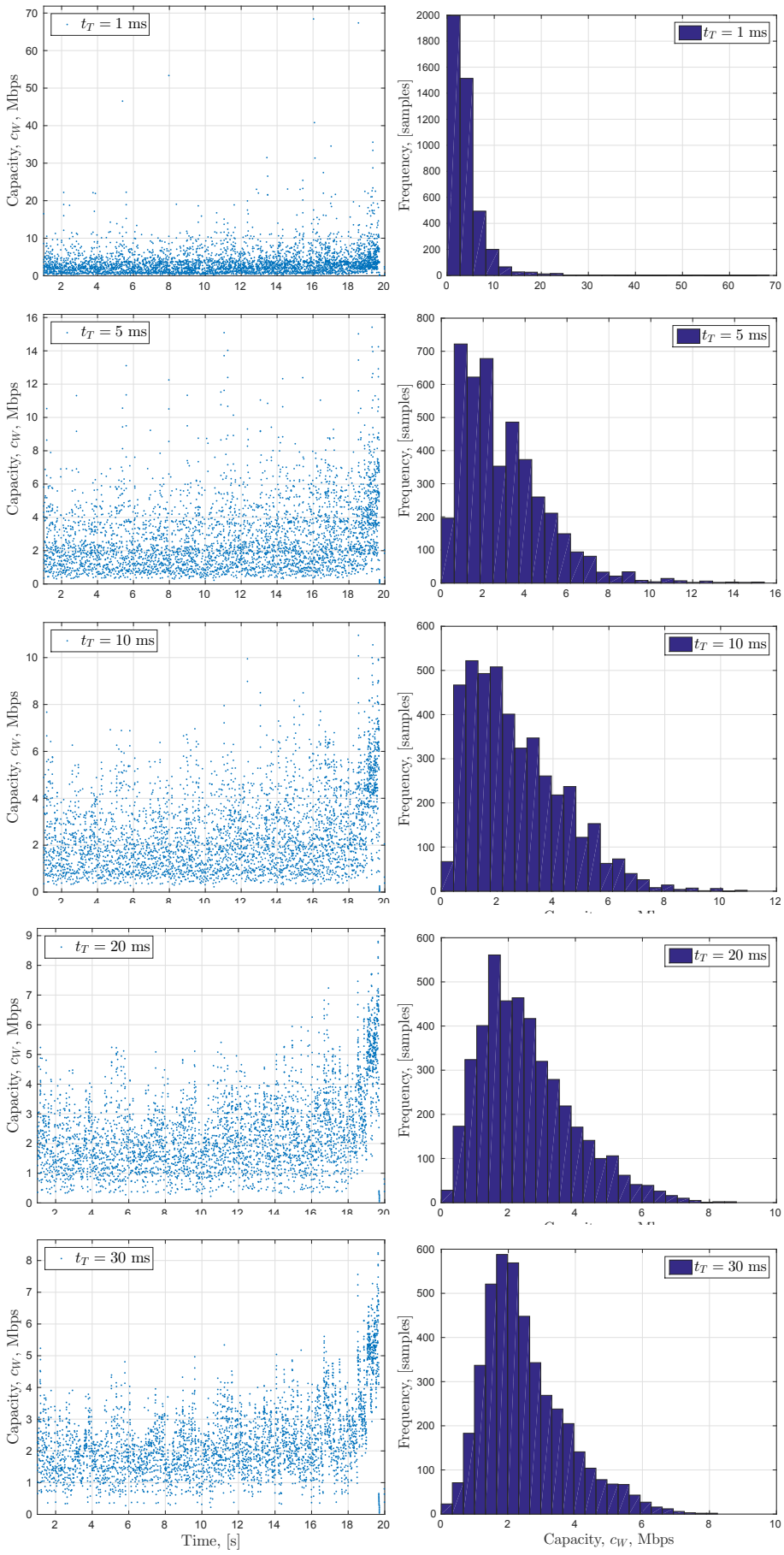


Figure 7.2: Scatterplots of c_W (left of each pair) and its statistical distribution (right of each pair) computed for $t_T = \{1, 5, 10, 20, 30\}$ ms from left to right. When the dispersion time is computed on windows larger than the TTI, $t_T > t_S$, the distribution gets more stable.

user is going to have in the future and is usually the bottleneck.

Figure 7.1 illustrates the packet dispersion due to the transmission over links at different link capacities. This example is based on LTE, but similar effects are observed in various mobile technologies. Initially, (1) the server sends a burst of IP packets (A-H in the example) back to back. The number of packets in the burst varies since it depends on a number of factors like the state of TCP connection, the specifics of the application and the server that generates it. Subsequently, (2) the base station (eNodeB) receives the packets, which have suffered variable delays due to the different link capacities and cross traffic encountered along the path. When the scheduler allocates a TB (marked with horizontal arrows in the plot) to the receiving UE (3), as many packets as possible are encapsulated in it. Therefore, all the packets that are scheduled together arrive within the same TTI at the UE. As a consequence, the inter-packet interval can be greatly reduced (packets A and B) or greatly magnified (packets B and C).

Considering the set of "back-to-back" transmitted packets crossing the path in Figure 7.1, we can distinguish their arrival rate R_A at the antenna from their transmission rate from the antenna to the user, which can have a maximum value of C_U. Both metrics are dynamic and are affected by the same parameters that affect R. Thus, if we sample them for a specific period of time, we may notice the following relationship between them. If $R_A > C_U$, the set of packets arrives at the BS with a delay which is inversely proportional to R_A and shorter than the average time needed for the BS to serve all but the last packet. Since the arrival rate is higher than the departing rate at the base station, the dispersion of the set is caused by the last link. Also, depending on the scheduling strategy, the set may be served within the same transport block or multiple transport blocks by the BS. Conversely, if $R_A < C_U$ the set of packets arrives at the BS separated by a delay which is longer than the average serving time of the BS. We thus have three cases (excluding the problematic cases presented in Chapter 5):

I Bursty arrival [16, 46] (e.g.: set of packets E-F), if $R_A > C_U$ and packets are in the same transport block.

II Wireless link capacity, if $R_A > C_U$ and packets are in different transport blocks (e.g.: set of packets A-D).

III The bottleneck link being in the server-BS path and/or the server transmitting at a very low rate (e.g. TCP slow start), if $R_A < C_U$.

In order to estimate C_U, we have to filter both *i*) and *iii*) cases, as well as take into account the behavior of sets of packets when transmitted over mobile networks as presented in Chapter 5. In brief, our approach has two components: a) generating capacity estimation samples which are not significantly affected by the above and b) the statistical processing of those samples in order obtain a C_U value.

7.1.1. Capacity Estimation Samples

The input data for our passive measurement tool are the timestamps and sizes of all the received data packets of a smartphone. We ignore packets related to connections establishment such as TCP and TLS handshakes, since they cannot saturate even momentarily the wireless link. This information can be collected on the OS level by monitoring the stack. In our experiments, we use rooted Android smartphones and tcpdump to capture all the incoming traffic. Ultimately this functionality could be included in the mobile OS as an on-demand lightweight measurement service.

We consider a set of N packets sent from a server and received at the UE so that the i-th packet is received at time t_i, with $i = \{1, \ldots, N\}$. A key metric used by our algorithm is the "inter-packet interval", the time difference between the arrival of two consecutive packets $(t_{i+1} - t_i)$. Obviously, in a group containing N packets, there are $N - 1$ intervals. W represents the unitless number of such intervals that we take into account when we generate the capacity estimation samples. For each packet in the set we define the dispersion time $d_W(i) = t_{i+W} - t_i$, and the per user capacity sample $c_W(i) = (\sum_{j=i}^{i+W-1} L_j)/d_W(i)$, for a given value of W, where L_i is the length of i-th packet.

In detail, the $c_W(i)$ value of packet i is derived by adding the sizes of W consecutive packets, starting from i and then dividing by the time duration of W consecutive inter-packet intervals, starting from $[t_{i+1} - t_i]$. Packet $i + W$ contributes only to the denominator. For example, in Figure 7.1, $c_{W=2}(A)$ is computed by dividing the sum of sizes of the packets A and B by the dispersion time $d_{W=2}(A) = t_C - t_A$.

The three arrival cases above contribute to the distribution of the capacity samples in different ways. Arrivals of type *i*) cause a tiny d_W and, thus, skew the distribution to the right (over-estimation of C_U). At the same time, type *iii*) events, which show larger d_W (under-estimation of C_U) skew the distribution towards the left. To better visualize what is discussed next, Figure 7.2 shows a set of scatterplots of c_W and histograms of its distribution computed on a single download performed using the Speedtest application [45] over a HSPA connection. The X-axis of the scatterplots represents the arrival time of packet i and the Y-axis its c_W value.

The impact of type *i*) arrivals can by limited by setting W appropriately. The idea is to include in each measurement packets belonging to different TBs in order to make sure that the highest throughput c_W we can measure is only related to the cell capacity and not to bursty packet arrivals, as it would have happened had we chosen $W = 1$ in the example of Figure 7.1. In order to achieve that, it is sufficient to study groups that, starting from any packet i, contain W_i intervals so that the minimum dispersion time $d_W(i)$ is longer than the maximum TTI of the scheduler, abbreviated t_S:

$$W_i = \{\min(W) \mid \min_W(d_W(i)) > t_S\} \tag{7.1}$$

This guarantees that at least two packets within the W_i window are scheduled in two different transport blocks, since $t_{i+W_i} - t_i = d_{W_i}(i) > t_S$. In other words, we are averaging the burstiness

over two transport blocks. An effect of Equation (7.1) is that each packet i has a different W_i value, depending on the spacing of packets that were received after it.

It is important to select the minimum value of W for the creation of the $c_{W_i(i)}$ value for packet i that has the property $\min(d_{W_i}(i)) > t_S$. As discussed in Chapter 5, the "slow start" behavior of TCP introduces noticeable gaps in packet delivery. Thus, samples that include these gaps in their calculation of d_W, generate c_W values that are significantly smaller and not representative of the C_U. A high value of W increases the probability of a sample to include such gaps.

7.1.2. Statistical Processing of the Samples

Now that type *i*) events are filtered, we ensure that each set spans across at least two TBs. The minimum dispersion time $\min d_{W_i}(i)$ for every packet i of the flow cannot be smaller than the minimum time needed for a set of packets to cross the wireless link, which corresponds to the maximum per user cell capacity. Thus, C_U can be found as the maximum of the distribution of c_W, which is equivalent to the maximum value of c_W.

$$C_U = \max_{i \in [1,\ldots,P]} c_{W_i}(i) \tag{7.2}$$

P is the total number of data packets of a flow. Note that, with Equation (7.1) we are filtering the effect of type *i*) arrivals ($\min$) and with Equation (7.2) the delays introduced by type *iii*) arrivals ($\max$).

Ideally, we would like to sample c_W until its distribution is stable, but C_U is varying because of both user movements and fast fading. Hence we can only obtain an estimate ${C_U}^{(p)}$ of it from a set of p consecutive estimation samples, where $p < P$. Although estimating the distribution from a limited number of samples reduces the accuracy of our measurement, we can at least guarantee that we are not overestimating C_U:

$${C_U}^{(p)} = \max_{i \in [1,\ldots,p]} c_{W_i}(i) \leq \max_{i \in [1,\ldots,P]} c_{W_i}(i) = C_U \tag{7.3}$$

This follows from the probability of the distribution of a sampled random process to contain the maximum of the theoretical distribution of the process, which is increasing with the number of collected samples:

$$\lim_{p \to \infty} {C_U}^{(p)} = C_U \tag{7.4}$$

7.1.3. Capacity Measurement

This section describes the feasibility of lightweight active and passive measurements of per user capacity C_U based on dispersion samples of packet sets. It also explores the effect different values of some parameters have on our technique. We compute the dispersion time by using an

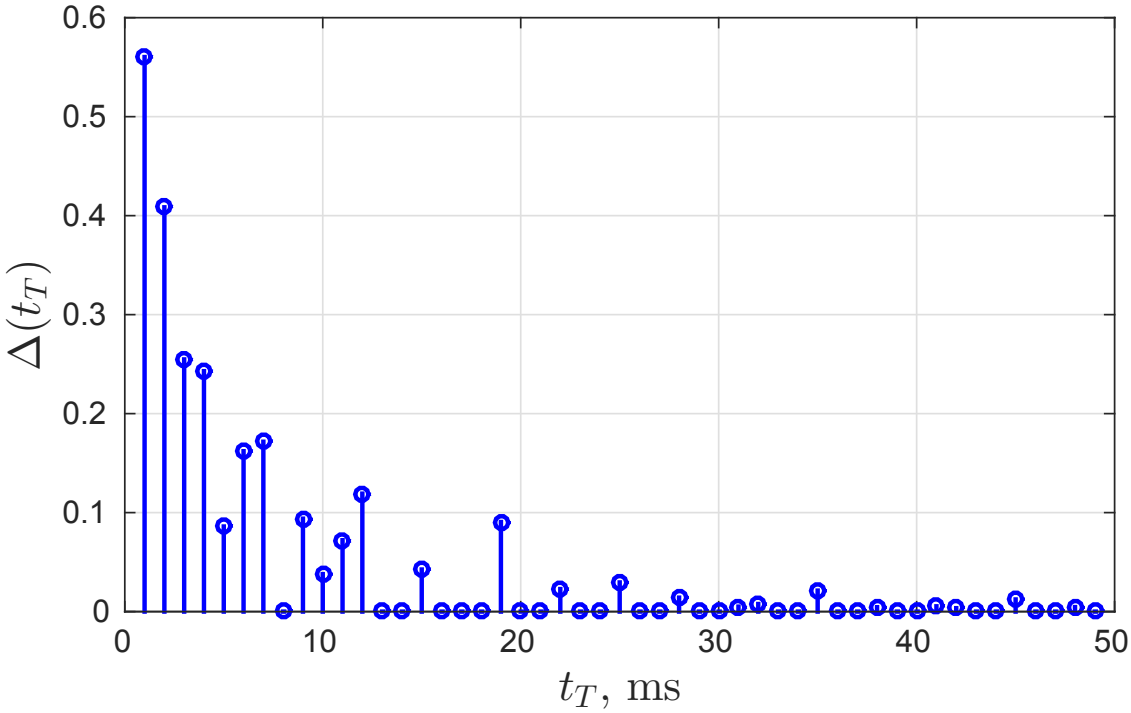


Figure 7.3: Ratio $\Delta(t_T)$, varying $t_T \in [2, \ldots, 50]$ ms. The measurements get stable from $t_T > t_S = 10$ ms.

adaptive window W_i intervals long for every packet i such that:

$$W_i = \{\min(W) \mid t_{i+W} - t_i > t_T\}, \tag{7.5}$$

where $t_T \in [1, \ldots, 50]$ ms, for all the values of t_T. The estimation sample of the i^{th} packet is composed of all packets following i until the first packet which arrived at least t_T ms later than i. This allows to satisfy Equation (7.1) a posteriori if the TTI duration is not known.

We exemplify the dispersion time in Figure 7.2 based on data obtained by time-stamping the arrival time of the packets of a 6 MB HSPA download. The figure presents the evolution of the scatterplots of c_W and the corresponding histograms of the c_W distribution for various characteristic values of t_T.

During the slow start phase of a TCP connection an increasing number of packets are sent back to back from the server, and after a few RTTs the congestion window is large enough to allow the transmission of packet trains long enough to measure capacity as high as 100 Mbps. In fact, C_U should be proportional to the maximum number of packets that can be scheduled in a single transport block and, if Equation (7.1) is satisfied and $t_T > t_S$, the impact of outliers due to bursty arrivals is removed. With reference to Figure 7.2, it can be seen that the maximum of c_W is approaching a stable value of about 10 Mbps when $t_T \geq 15$ ms. A similar analysis on the rest of our dataset reveals that a stable value is reached for values of t_T between 10 and 20 ms.

Moreover, Figure 7.3 shows the stability of the maximum of the capacity by plotting the ratio

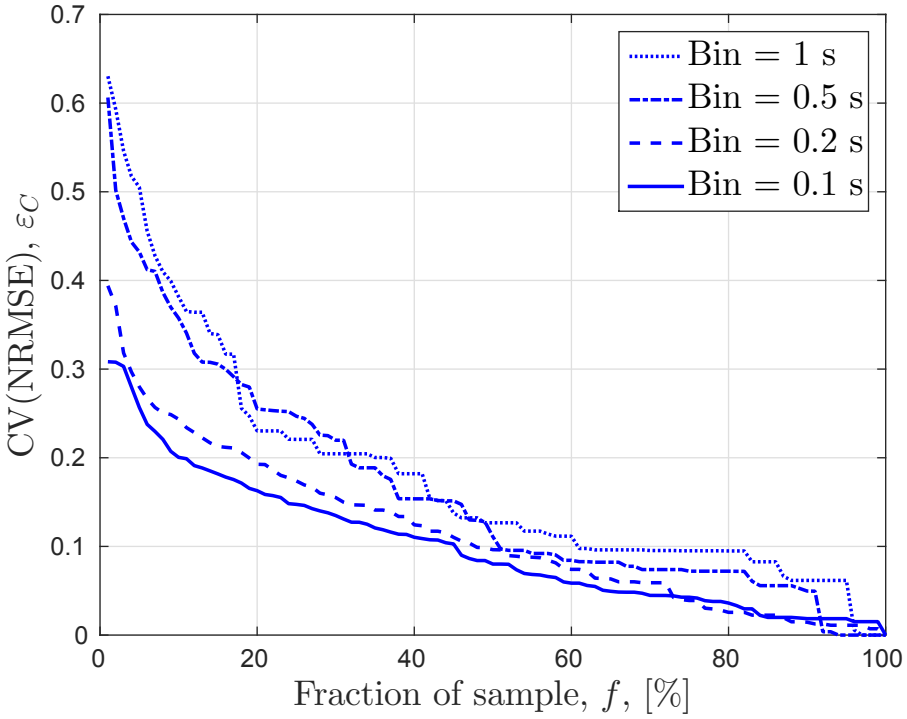


Figure 7.4: Coefficient of variation of the normalized root mean square error ε_C of the capacity estimate computed over a fraction $f = k/K$ of continuous samples for varying bin sizes ($\{0.1$s, 0.2s, 0.5s, 1s$\}$).

$\Delta(t_T)$, computed between the maximum value obtained with windows of $[t_T]$ and $[t_T - 1]$:

$$\Delta(t_T) = \frac{|C_{W|t_T} - C_{W|t_T-1}|}{C_{W|t_T-1}} \tag{7.6}$$

Ideally, the ratio $\Delta(t_T)$ should stabilize to 0 as soon the scheduling outliers are filtered ($t_T > t_S$) and further increasing t_T should only make the distribution smoother. However, in actual experiments increasing t_T makes it more difficult to obtain a sample of the maximum capacity which is consistent over different transport blocks. In this preliminary example, we can see that $\Delta(t_T)$ becomes stable for $t_T > 20$ ms, which is in line with the HSPA TTI of $2 - 10$ ms.

Next, we divide the time duration of a download into fixed sized bins. We apply the above method taking into account only a percentage $f = k/K$ of consecutive capacity samples in each bin. In this case, K is the total number of samples inside each bin and k is the number of consecutive samples that we consider for every bin. Figure 7.4 shows the coefficient of variation of the normalized root mean square error – CV(NRMSE) – of the estimate ε_C, by varying f:

$$\varepsilon_C = \sqrt{\frac{\sum_{\text{bins}}(C^{(k)} - C^{(K)})^2}{N_b \mathrm{E}[C^{(K)}]^2}}, \tag{7.7}$$

where N_b is the number of bins in a flow. The computations have been repeated for different bin

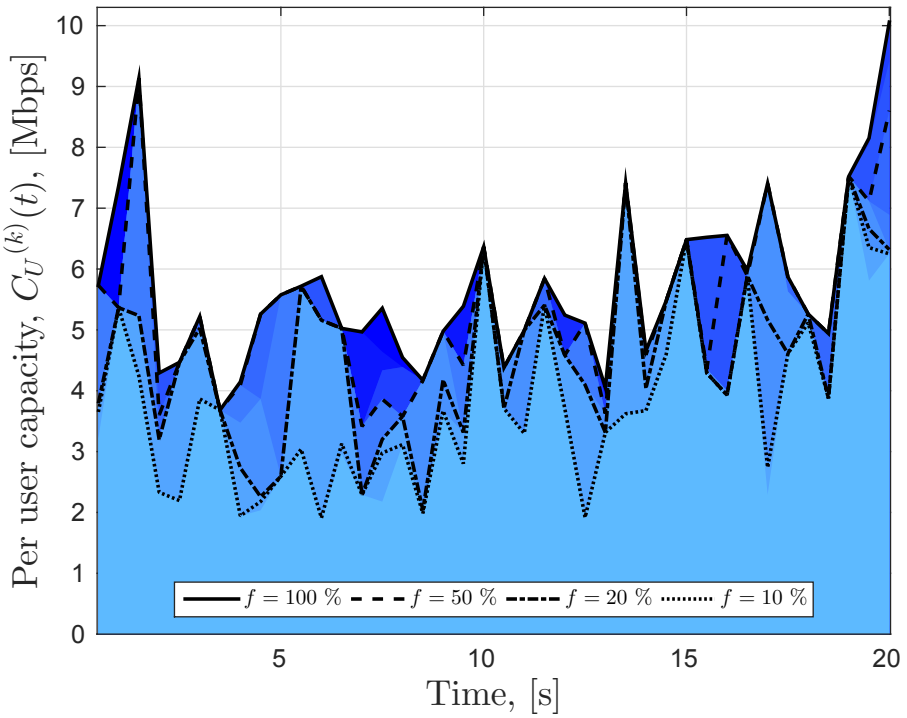


Figure 7.5: Time plot of the capacity variation $C_U{}^{(k)}(t)$ computed every 500 ms and its different estimates computed with $f = \{10, 20, 50, 100\}$ %.

sizes varying in $\{1, 0.5, 0.2, 0.1\}$ seconds (dotted, dash-dotted, dashed and solid lines, respectively). It can be seen that the error decreases below 20 % when more than 20 % of the samples are used.

Figure 7.4 can also be interpreted as the width of the probability distribution of having an exact measurement using f % of the samples. In particular, it is easy to see that when we use all the samples, the distribution should collapse into a delta function (zero width), while the fewer samples we use, the wider the distribution. The real value can only be larger than the measured one, because of Equation (7.3) that shows $\max_{i\in[1,\dots,k]} c_{W_i}(i) \leq \max_{i\in[1,\dots,K]} c_{W_i}(i)$. Thus, this distribution has non-zero width for values smaller than the actual measurement only.

To complete this preliminary evaluation of our measurement technique, Figure 7.5 shows the variation of the per user capacity $C_U{}^{(K)}(t)$ measured every 500 ms and its estimates $C_U{}^{(k)}(t)$ computed with $f = k/K = \{10, 20, 50, 100\}$ % (dotted, dash-dotted, dashed and solid lines, respectively). Although with 10 % of samples the estimates are quite different from the actual capacity values, we will be showing next that it is possible to exploit these coarse estimates to obtain a sufficiently accurate capacity estimate.

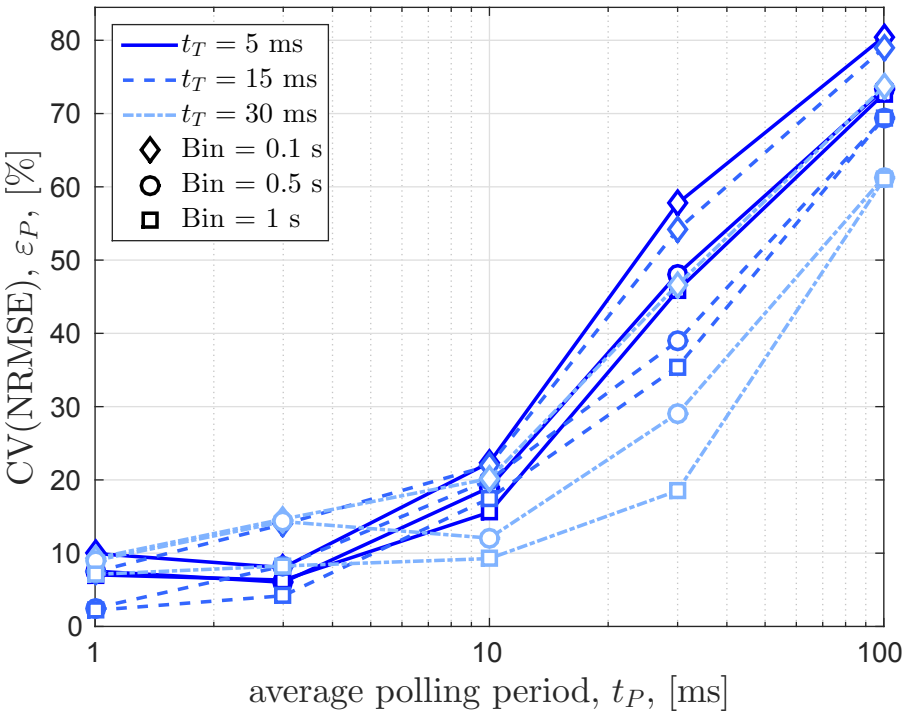


Figure 7.6: CV(NRMSE) ε_P of the capacity estimate between ideal arrivals ($t_P = 0$) and arrivals that suffer from polling ($t_P \neq 0$), for varying bin sizes and minimum dispersion times t_T.

7.2. Simulation Campaign

We have performed an extensive simulation campaign in order to evaluate our proposed technique in a controlled environment. We use a modified version of ns-3.23 [76] and its LTE module LENA [77]. We focus the simulation part of this study on LTE due to its increasing popularity. In all simulations the monitored user uses TCP, since it is both the most challenging and the most popular [16] transport layer protocol of mobile phones. The variable parameters of the simulations are presented in Table 7.1. The fixed parameters are: 1) the simulation lasts for 22 seconds and 2) the BS uses a proportionally fair scheduler. For each set of parameters we run the simulation multiple times with a different seed, generating in total 18570 flows.

Table 7.1: Simulation parameters

Parameter	Value
Number of resource blocks (Mhz)	25 (5), 50 (10), 75 (15), 100 (20)
Number of competing UEs in the cell	$[0, 1, 2 \ldots, 10]$
Distance between UE and BS in m	$[0, 50, 100 \ldots, 450]$
Number of interfering BS	$[0, 1, 2 \ldots, 6]$
Type of scenario	“Static”, “Urban walking”, “Vehicular”

Next we investigate the effect of polling on the accuracy of the measurements. The simulation results do not suffer from polling, thus the packet arrival time reported in the logs is the actual

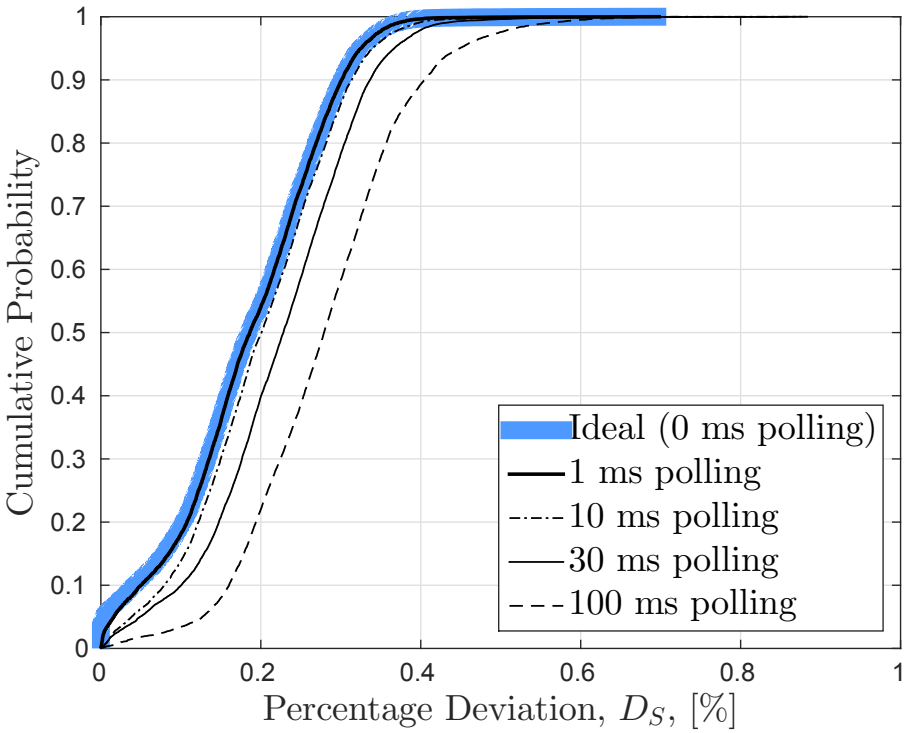


Figure 7.7: Deviation of the sampling estimations ($k = 5\%$) for various average polling periods t_P from the ideal case ($k = 100\%$, $t_P = 0$).

arrival time at the NIC. In order to simulate the polling effect we manipulate the logs so that we check for incoming packets every $t_P \pm 10\%$, where $t_P \in [1, 3, 10, 30, 100]$ ms. We add the 10% deviation in the timing of each polling because based on our traces and the literature, polling does not have a fixed frequency. We also add a tiny inter-packet delay (in the range of 0.1 ms) between the packets that are reported together by the polling function, in a fashion similar to the one we observe in our "real life" traces. Please note that the polling delay (if present) is usually within 10 ms under normal circumstances.

Figure 7.6 shows the CV(NRMSE) ε_P between traces that have the original timestamps and processed ones. We calculate the ε_P as we did for the ε_C in Equation (7.7).

$$\varepsilon_P = \sqrt{\frac{\sum_{\text{bins}} (C^{(t_P)} - C^{(0)})^2}{N_b \mathrm{E}[C^{(0)}]^2}} \tag{7.8}$$

It can be seen that the error is at most 20% for most cases (up to 10 ms of delay).

Subsequently, we examine how the combination of sampling only 5% of the available estimators and polling affects the accuracy of the results. We divide every flow to 100 ms bins and for every bin we calculate the ${C_U}^{(100\%)}$ and the ${C_U}^{(5\%)}$ for various t_P values. The speed of each flow is the average of the measured capacity of all its bins $\mathrm{E}[{C_U}^{(k)}]$. As a groundtruth, against which we compare the rest of the results, we suppose the case where $t_P = 0$ (ideal polling) and $k = K$. Figure 7.7 depicts the Empirical CDF of the percent Deviation D_S computed by the

formula:

$$D_S = \frac{|\mathrm{E}[{C_U}^{(5\%)(t_P)}] - \mathrm{E}[{C_U}^{(100\%)(0)}]|}{\mathrm{E}[{C_U}^{(100\%)(0)}]} \tag{7.9}$$

By comparing the ideal line of $t_P = 0$ with the rest, we conclude that even though polling does have a negative effect in the measurements, the dominant cause of error is the sampling. Also, we observe that for the most common t_P values ($t_P < 10$ ms) the deviation for 90% of the cases is less than 30%.

7.3. Measurement Campaign

In order to validate our measurement technique over many different "real life" scenarios and configurations, we organized a measurement campaign that covers two cities in two different countries, Darmstadt (Germany) [26] and Madrid (Spain), for 24 hours a day lasting 7 days. During this time, 5 people per city moved around as they normally do, carrying one measuring device each and performing their usual tasks involving mobile networking on the measuring devices. In order to be able to compare results of both passive and active measurements, we also perform automated periodic file downloads.

All the devices were running a simple Android application, which was periodically sampling the available capacity by starting two download types: *short* downloads of 500 KB to study the TCP slow start phase and *long* downloads of 2 MB to measure TCP steady state throughput. The two types were organized in a sequence with a long download, preceded by two small downloads and later succeeded by another two. We use tcpdump on the measurement devices to monitor the arrival time and size of all incoming packets. The download sequence was repeated every 50 minutes. Additionally, we log other related phone parameters: GPS, cell ID, Channel Quality Indicators (ASU, dBm) and network technology (2G, 3G, LTE).

The phones used in the campaign were the following: 5 Nexus 5, located in Germany, and 4 Sony Xperia Miro and 1 Samsung Galaxy S3, located in Spain. Also, while the Nexus 5 phones are LTE capable, the other phones only support radio technologies up to HSPA.

7.4. Results and Discussion

We verified our measurement technique by analysing more than 3000 unique TCP flows extracted from the communication of the phones participating in the campaign. As before, we split each flow into 100 ms bins and calculate the ${C_U}^{(100\%)}$ and ${C_U}^{(5\%)}$ metrics, and assume that their average is the speed of each flow. Note that in these measurements we neither have control over the polling, nor we can distinguish it from the scheduling behavior.

Figure 7.8 shows a scatterplot where the abscissa and the ordinate of each rectangular point are the sampled and non-sampled versions of C_U, respectively. Further we add in the same plot the related simulation results for $t_P = 3$ ms as diamonds. As expected from Equation (7.3) all the

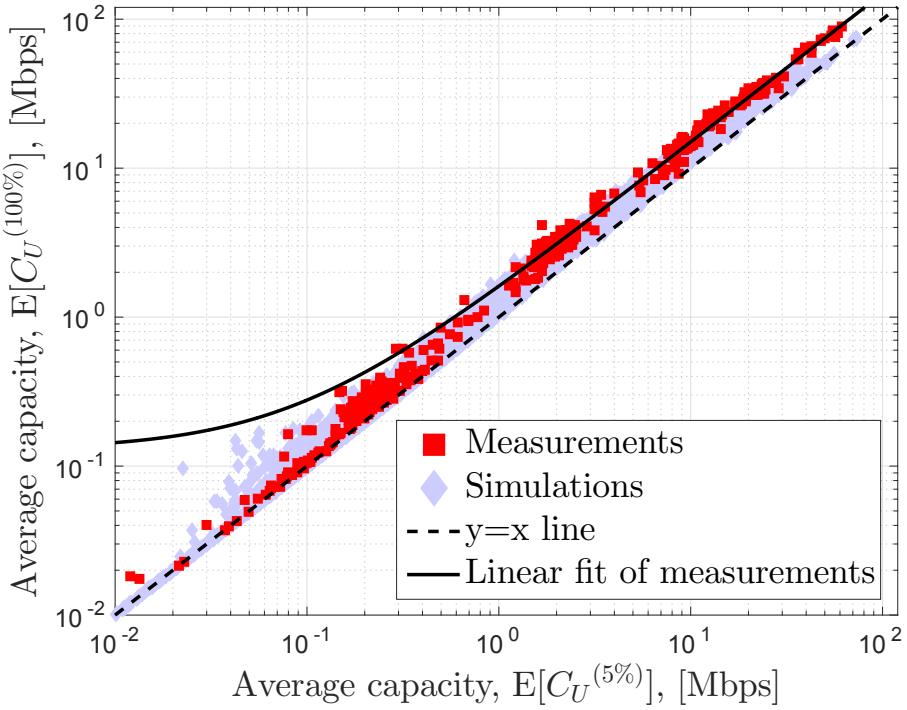


Figure 7.8: Scatterplot of the average estimate of per user capacity computed using all available information $\mathrm{E}[{C_U}^{(K)}]$ against the estimate computed 5 % of the available information $\mathrm{E}[{C_U}^{(k)}], k = K/20$.

data points are above the $y = x$ line. Thus, we verify that our algorithm may only underestimate the capacity. The fact that all the points are so close to the $y = x$ line proves that the values derived by just 5% of the samples are good estimators of ${C_U}^{(100\%)}$. As a consequence, this measurement can be safely used as a lower bound in resource optimization problems. We also plot the linear regression of only the actual measurement results as a dashed line. The regression line would allow us to build an even better estimator with lower error.

The figure is plotted in double logarithmic scale in order to emphasize that the relationship between ${C_U}^{(100\%)}$ and ${C_U}^{(5\%)}$ can be observed over all the measured connection rates and there is an almost constant ratio between the estimate and the actual value. Although outliers are visible, we can obtain quite an accurate estimate of C_U by exploiting as few as 5 % of the packets sent during a TCP connection. This allows for quite an effective passive monitoring technique as, even by monitoring small data exchanges, it is possible to obtain frequent and accurate mobile per user capacity measurements necessary for user throughput prediction and resource allocation. The linear regression line seems to deviate from the measurement "cloud" for low values of capacity, because of the double logarithmic scale used in the plot, which highlights the regression offset for low values (500 Kbps and less). Further, we observe that for high values, the regression line has an almost fixed vertical distance from the $y = x$ line (constant percentage error). This represents the error of the estimate and, since it is constant, in the double logarithmic plot, appears as a fixed deviation on the Y-axis from the $y = x$ line.

Unfortunately, using very low rate background traffic is impossible. The rates of such traffic are on the order of 4 packets over 100 ms, which do not allow for reliable capacity measurements. Also, a big number of the APPs use the Google Cloud Messaging (GCM) service, which minimizes their notification related traffic. In the case of GCM, if there is an update a few packets are sent just to generate a notification. When the user interacts with the notification, a larger number of packets are downloaded. In this scenario, we can use that download to get an estimation.

In the experiments, we use rooted Android phones and tcpdump to perform the measurements. Given the very low complexity and resources that are required by our approach, the C_U estimation is generated at virtually no cost. Therefore, we believe that it may be included in the OS as a service to applications that may opt-in to use it. For example, the flow-id, the timestamp and the size of a packet could be registered as part of the standard kernel packet processing procedure. Since these values do not contain any sensitive information, there are no privacy concerns and after a short period to time, when this information is irrelevant it can be deleted. Upon application request, the OS could generate a C_U estimation, if there are sufficient data stored. The knowledge of the flow-id can help distinguish the state of a TCP flow (slow-start, steady-state etc.). If it is possible to use small values of t_T, it is possible to generate accurate estimators even during the late part of slow start, when the congestion/receive windows have relatively high values, since then the dispersion time can be smaller than the time required by the antenna to transmit a server burst. In case of a TCP flow that stops very early, it can be difficult to remove both the slow start and the scheduling artifacts. In such cases, the resulting value will be significantly lower than the truth, but this is easy to detect and filter (e.g., requiring a flow to generate at least 75 downlink packets in order to be used).

As a side note, our technique is also able to estimate fast per user capacity variations. However, it obtains a lower accuracy since a larger fraction of samples are needed to estimate the maximum of the c_W distribution. Nonetheless, it is often sufficient to use 20 % of the samples collected in a bin to achieve a reasonable estimate of C_U. In fact, with the smallest bin size and as few as 20 % of the samples have an error $\varepsilon_C < 0.2$, which means the actual capacity should not be larger than 120 % of the estimated value.

In addition, t_T must be taken slightly longer than the TTI to avoid the measurement being impacted by many bursty arrivals. In line with Equation (7.1) of Section 7.1, $\Delta(t_T)$ approaches zero for $t_T > 15$ ms for most of the recorded flows.

Figure 7.9 shows the CV(NRMSE) for various combinations of t_T and f of the measurement campaign flows. The bin size is set to 200 ms to give an example of this technique's results when it collects very frequent measurements. As expected ε_C decreases when t_T and f increase. For values of $t_T \geq 15$ ms and $f \geq 20$ %, the error is small enough for the model to give trustworthy results ($\varepsilon_C \leq 15$ %).

Finally, Table 7.2 shows some of the overall evaluation of the traces obtained by the measurement campaign with $f = 25$ % averaged over the bin size and using the optimal t_T ($\min t_T | \Delta(t_T) \to 0$). Optimal t_T and C_U are computed as described in Section 7.1 and then

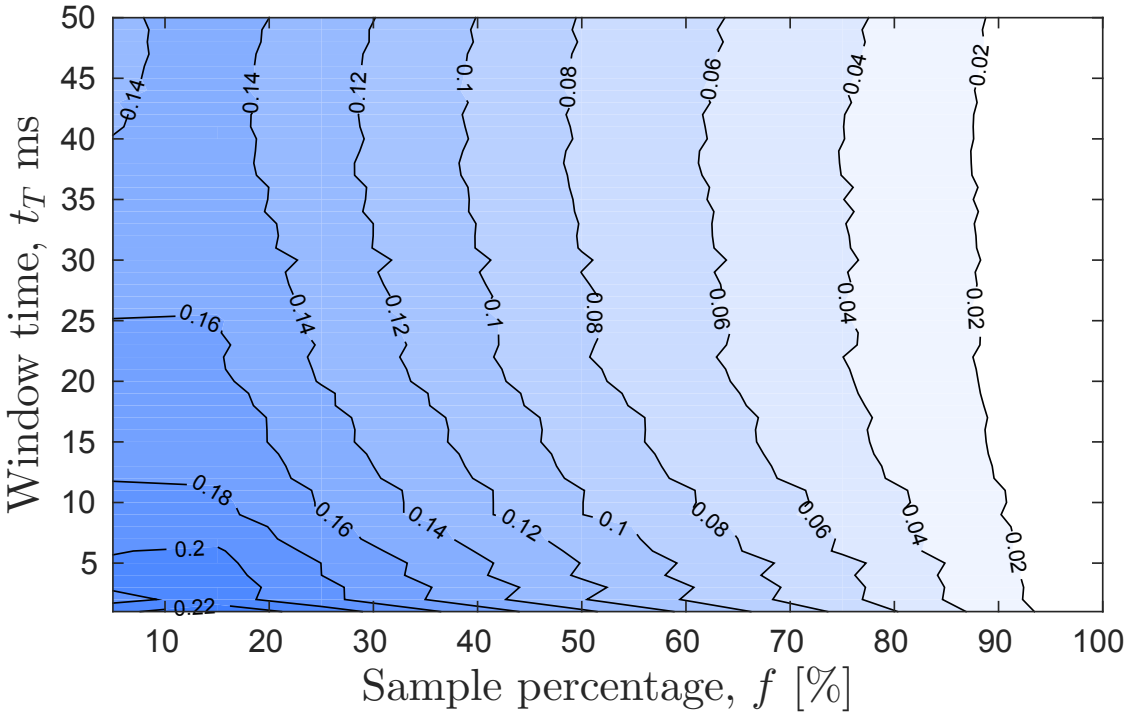


Figure 7.9: Contour graph of ε_C varying t_T and f for a bin size of 200 ms.

averaged over all the traces. While some of the flows are transmitted using 2G EDGE data, the results are not included since there are too few such flows for statistical significance.

Technology	UMTS	HSPA	HSPA+	LTE
C_U (Mbps)	10.83	1.4	10.74	24.3
Optimal t_T (ms)	19	23	17	16

Table 7.2: Average C_U and average optimal t_T per technology.

The measurements are based on the data reported by the Android OS. Note that HSPA and HSPA+ are a family of enhancements to UMTS, that greatly increase its speed. The high average speed of UMTS is related to networks that support the HSDPA enhancement for improved downlink speed, but not all the enhancements that would classify them as HSPA or HSPA+. The very big differences in speed between the HSPA, HSPA+ and LTE technologies can be explained by the following reasons. More recent technologies can achieve higher speeds. Smartphones tend to use the best technology possible for their channel quality. Thus, they use HSPA only when their signal is too bad to use a better technology and in turn the bad signal greatly affects speed.

Our approach is designed for downlink measurements, which account for the vast majority of the smartphone generated traffic [16]. Recent trends, though, show an increase in uplink related user activity and therefore we will briefly discuss the uplink case. Our algorithm cannot be directly applied to the uplink due to uplink communication characteristics. For instance, if we attempt to perform a measurement on the phone side we can gather very limited informa-

tion. Without accessing the transceiver firmware, we can only observe how fast packets appear in the kernel, instead of how fast the NIC successfully transmits them at the medium, which is the metric we are interested in. It is possible that packets may remain in the buffer of the NIC for a relatively long time after they appear in the kernel, leading to wrong estimations. On the other hand, applying our algorithm to measurements collected on the server side will fail to measure the cell capacity, since many intermediate hops may be between the eNodeB and the server. An alternative approach would be to infer clues of the speed indirectly at the phone side. If a UDP socket is blocking, it can be an indication that the rate at which an application is generating packets (which we can detect) is higher than the link capacity, thus deriving an upper limit of the speed. In the case of TCP traffic, the ACKs can be analysed to infer whether the rate that the application is generating traffic is above or below the link capacity. Further analysing the uplink scenario though is beyond the scope of this thesis.

7.5. Summary

We presented a lightweight measurement technique that leverages adaptive filtering over the packet dispersion time. This allows to estimate the per user capacity in mobile cellular networks. Accurate estimates can be achieved exploiting as few as 5 % of the information obtained from TCP data flows. We validated our technique over a week-long measurement and an extensive simulation campaign. We achieved good estimation accuracy even when using only short lived TCP connections. Since our technique is based on simple post-processing operations on the packet timestamps, it is possible to easily integrate it in background processes or OS routines. It is possible to extend our measurement application with filter based prediction capabilities, such as the ones proposed in Appendix A, in order to provide mobile phones with a complete capacity forecasting tool, which, in turn, may allow for advanced resource allocation mechanisms.

Part III

Interconnection of Third-Party Services and Mobile Operators

Chapter 8

A Measurement Study of Mobile Cloud Services

In the previous part we focused on how the multimedia experience can be improved based on solutions that work in the low level aspects of networking. In this chapter, we focus on management, configuration and political aspects of networking that have a measurable effect on user experience. More specifically, we are interested in how providers of cloud infrastructure and content delivery, collectively called CSPs, are interconnected with mobile ISPs and also measure how they perform. The prevalence of such services in mobile websites and applications, make them a key component of the mobile experience. If a CSP provider, especially one of the dominant players, underperforms, a big number of seemingly unrelated websites and services are affected.

In this chapter, we empirically analyse the web of relationships between mobile apps, CSPs, and MNOs. In particular, we aim to answer the following questions:

- *Which are the most dominant CSPs enabling the mobile Internet?*
- *How well are these CSPs interconnected with MNOs at a topological level?*
- *What is the performance of these services (*i.e.*, as perceived by end-users) when accessed from commercial MNOs?*

We conduct the first comprehensive study of its kind, combining different measurement techniques and vantage points to fully capture the synergies between the entities forming this complex ecosystem. As a starting point, we use traffic logs that we collected with Lumen Privacy Monitor [78], a mobile privacy and transparency tool. Lumen's rich traffic logs allow us to accurately identify the most prevalent CSPs providing on-line infrastructure to 8,281 mobile apps in the wild. Then, we run a purpose-specific month-long measurement campaign using the MONROE platform for mobile broadband measurements [79] to capture the interactions between ten commercial MNOs from four European countries and the most popular CSPs, as well as their transport- and application-layer performance. Specifically, we focus on analysing the effect of replica selection, the role of the DNS subsystem, and the impact of in-path TCP splitting proxies, as well as

routing- and peering-level effects on transport-layer performance. Our study also includes mobile subscriptions roaming internationally.

Our analysis reveals that six CSPs— Amazon Web Services (AWS), Google, Facebook, Akamai, Amazon CloudFront, and Highwinds — provide infrastructure and online support to 85% of the apps that we measure with Lumen. We capture interesting multi-CSP strategies that 687 second level domains (15% of domains) use to increase their geographical coverage and reliability (Section 8.3). We also track the integration and collaboration strategies between the top CSPs identified through Lumen and the MNOs available in the MONROE platform (Section 8.4). In particular, Akamai's strategic alliances with multiple MNOs stand out. The varying degrees of collaboration between MNOs and CSPs translates into notable performance differences, which we actively measure and analyse in the same section. Namely, the tight integration between MNOs and CSPs results in lower latency and connection times: Google's relationship with various MNOs has resulted in 15% lower connection times on average compared to other similarly performing CSPs. We detect various levels of EDNS adoption among the studied operators, which, however, does not seem to have any significant impact on performance. International roaming may add significant delays, especially in the case of well provisioned CSPs, defeating their attempts to put content close to the user. Finally, we observe that the choice of PoP may inflate by at least 20% the delay towards popular websites (Section 8.6).

8.1. Recent Trends

The line distinguishing a CDN from a cloud computing provider can be blurred at times: third-party service providers may simultaneously offer cloud computing and CDN services using the same domain names and IP blocks. Due to this classification challenge, in this study we analyse them together using the term *CSP*.

CSP deployment strategies: CDNs and cloud services may follow different strategies to deploy their servers at a global scale. Cloud services like AWS leverage a reduced number of datacenters located in strategic locations. Instead, most CDNs deploy thousands of caches and proxies as close to the end-user as possible to minimize the end-to-end latency. For example, Akamai's infrastructure controls more than 233,000 servers in over 130 countries and 1,600 networks [80], while AWS operates just 44 large-scale datacenters located in 16 geographic regions [81]. Large CDNs may also host their services in IP blocks owned by MNOs and fixed-line ISPs who may also commercialize their own CDNs and cloud solutions (*e.g.*, Level 3 and TeliaSonera). This state of affairs makes it difficult to attribute a given domain name or IP address to a particular CSP, as we will discuss in Section 8.4.

CSP-MNO integration: The quality of experience (QoE) the end user perceives when connecting to CSPs may be determined by the underlying connectivity agreements between MNOs and CSPs [67,68]. A recent crowd-sourcing measurement campaign [82] suggests that certain mobile domains perform poorly on many MNOs. App developers, MNOs and CSPs are increasingly

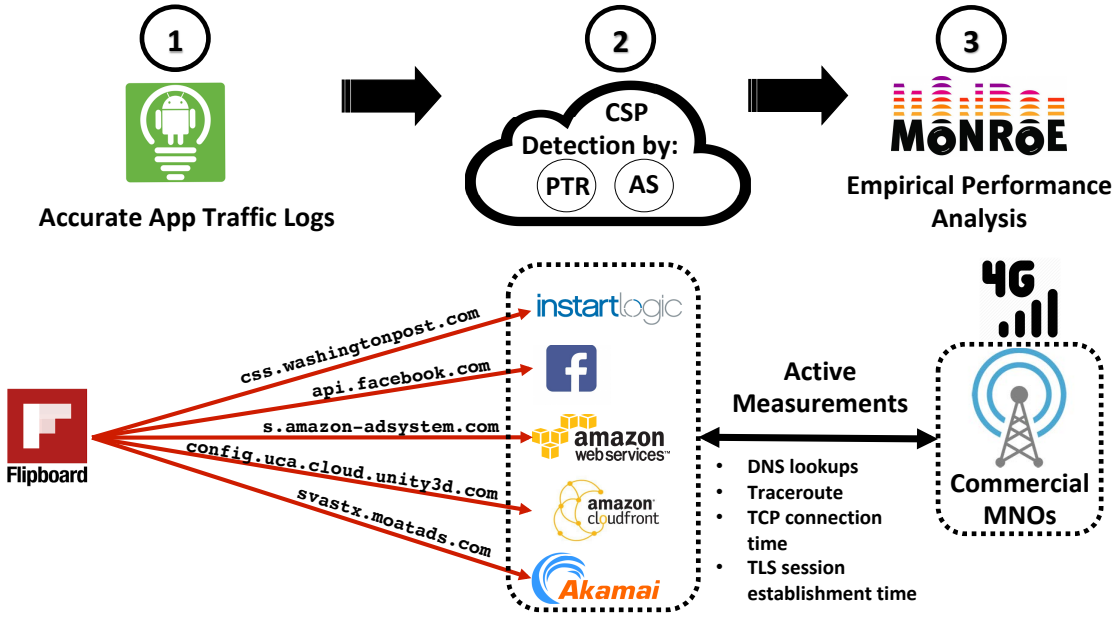


Figure 8.1: Schema of our study methodology using a simplified case of the Flipboard app as a toy example. We followed three complementary steps in our study: 1) we analyse app traffic logs to identify the network domains reached by thousands of mobile apps (each red arrow represents a traffic flow to a domain); 2) we detect those domains hosted in CSPs; and 3) we actively measure the performance of CSP-hosted domains on the MONROE measurements platform.

engaging in new peering agreements [67, 83, 84] and initiatives to avoid such inefficiencies. Two examples are Netflix' Open Connect Initiative [85] and Akamai's Accelerated Network Partner (AANP) program [86].

Multi-CDN strategies: CSP usage by mobile apps can be complex at times. Some apps combine several cloud services to perform specific operations – *e.g.*, Netflix uses AWS for encoding their videos while using multiple CDN providers to enhance their resilience, coverage, and efficiency. This strategy is known as *Multi-CDN* [87]. Adhikari *et al.* [87] studied Netflix and Hulu's multi-CDN strategies and their CDN selection algorithms. Their work concludes that considering network conditions in the CDN selection algorithm or utilizing multiple CDNs simultaneously can improve the average available bandwidth by 12% and 50%, respectively.

8.2. Methodology and Datasets

We follow a multi-step research method to study mobile CSPs as we depict in Figure 8.1. In summary: i) we obtain accurate mobile traffic traces provided by thousands of users of the Lumen app to identify the set of network domains reached by thousands of mobile apps; ii) we rank each domain by its popularity, and identify those hosted on CSPs using a purpose-built CSP classifier; iii) we measure the performance of CSP-hosted domains on the MONROE platform and infer peering relationships between CSPs and MNOs. As our study is built upon real-world

mobile traffic, we can comprehensively study the most prevalent CSPs by running realistic active measurements on a set of representative CSP-hosted domains.

Toy example: the Flipboard app. Figure 8.1 provides a high-level description of our method using the Flipboard app as a toy example, depicting how our method helps us understand the relationships between mobile apps, domains (specifically, Fully Qualified Domain Names (FQDNs)), and CSPs hosting these domains. First, we use the Lumen Privacy Monitor (Lumen) [78] to capture the different domains Flipboard connects to during normal operations (**Step 1** in Figure 8.1). In **Step 2**, we combine a number of techniques — including reverse DNS lookups, domain classification, and IP block analysis, among others — to identify which domains rely on cloud providers, and to determine the actual CSPs providing support. This step allows us to know that the Flipboard app communicates with 5 different FQDNs, and that each contacted domain is hosted in a different CSP.

Mobile apps, including Flipboard, typically connect to third-party services for purposes of advertising and tracking [88], or to embed other services like online payment and weather reports [89]. These third-party services may also rely on CSPs for outsourcing their cloud infrastructure. For example, the Flipboard app leverages Facebook's Graph API, which is hosted in Facebook's own cloud infrastructure, for user login and possibly for advertising purposes. Armed with this FQDN-CSP mapping, we select a number of representative domains to perform active performance measurements on (*e.g.*, TCP connection time), using the MONROE platform (**Step 3**). We further describe each step and their relevant datasets in the following subsections.

8.2.1. Step 1. Collecting Accurate Traffic Logs

Lumen is a free Android tool for transparency and user control that captures, reassembles, and analyses mobile app's traffic flows on the device itself. Lumen operates as a middleware between apps and the network interface, and intercepts all network traffic locally and in user space using the Android VPN API. This allows Lumen to correlate traffic flows with disparate and rich contextual information available on the device. For example, Lumen matches DNS queries to outgoing flows and the app process owning the socket in order to obtain an accurate profile of a given app's traffic patterns.

Lumen is publicly available to download from the Google Play Store [90], allowing us to crowd-source mobile traffic measurements at scale from all over the world. This feature makes Lumen a unique mobile vantage point to understand how mobile apps communicate with online services using real user input and network-stimuli and, therefore, the real interactions between mobile apps and the CSPs supporting them. Lumen's global user base allowed the collection of a representative dataset accounting for over 5M anonymous network flows corresponding to over 8,000 different mobile apps reaching more than 18,000 FQDNs. In order to preserve user privacy, Lumen performs its flow processing and analysis on the device, only sending anonymized data —

no payload or user identifier is collected – to our servers[1].

8.2.2. Step 2. Mapping FQDNs to CSPs

Identifying the synergies between mobile apps, FQDNs, and CSPs is a challenging problem. To tackle this problem, our approach focuses on the 18,000 FQDNs available in the Lumen dataset. We retrieve and analyse the PTR records (if available) associated with each IPv4 and IPv6 address by running reverse DNS lookups to identify whether a FQDN is hosted on a given CSP. This allows us to map FQDNs to CSPs using CDNFinder's PTR to CSP mapping [91].However, CDNFinder's mapping does not include marginal CDNs like CDNetworks as well as pure cloud service providers like AWS or Claranet. Moreover, we could only retrieve PTR records for 62% of the total IP addresses present in the Lumen dataset. In order to overcome these limitations and increase CDNFinder's coverage, we take the following steps:

1. We run a semi-supervised PTR classification by searching for strings that may suggest CSP-related operations like ``cdn'' and ``host'' on the PTR records.

2. To identify CSP-related PTR records that are absent in CDNFinder's mapping, we implement a semi-supervised PTR classifier that leverages public domain classifiers, specifically McAfee's [92] and OpenDNS' [93] domain classifiers. To that end, we first extract the most common categories assigned by the aforementioned domain classifier services to well-known CSP-related PTR records (namely "Internet Services" and "Content Server"). Then, we check if any of the PTR records that we obtain through our reverse DNS lookups fall in any of these categories. Unfortunately, this approach introduces false positives as third-party in-app services like ad networks may be classified as "Internet Services" too. The sheer size of the PTR records impedes our ability to sanitize all of them manually so we limit our manual inspection to PTR records associated with 248 popular FQDNs.

3. For each IP address associated with Lumen's FQDN entries, we run WHOIS queries and retrieve the information on registrant organization and listed email addresses. We browse the website of the email address domain to check if the organization offers any CSP-related products.

4. To identify CSPs in IP addresses that do not have PTR records associated with them, we leverage the organization name string as present on AS-level records. This analysis allows us to infer the presence of CSPs for 37% of FQDNs without PTR records and to increase the identification coverage of FQDNs associated with CSPs by 14%.

Combining these four techniques allows us to create a mapping of 194 second-level PTR records associated with 125 third-party CSPs, of which only 43 were initially present in CDNFinder. We made our PTR- CSP mapping open to the public [94].

[1]Our institutional IRB classifies this project as "non-human research subject" as we analyse the behavior of software, and not people.

Table 8.1: List of MNOs per country. MNOs listed in bold are roaming internationally (home country code in brackets).

Country	MNOs
Norway	Telenor, Telia, **Telia (SE)**
Italy	Vodafone, Wind, TIM
Spain	Yoigo, Orange, **Vodafone (IT)**
Sweden	Telia, Telenor, 3

Summary: Our extended FQDN-CSP mapping — both for IP addresses as well as `PTR`-records — and their associated AS numbers allow us to measure the prevalence of each CSP in the mobile ecosystem, reveal instances of multi-CDN strategies, analyse CSP peering relationships with MNOs, and compile a set of representative domains to empirically measure on the MONROE platform (**step 3**). Due to time and technical restraints, we limit our active measurements campaign to a subset of 1,334 FQDNs hosted by the 6 most prevalent CSPs across apps: Amazon Web Services (AWS), Google, Facebook, Akamai, Amazon CloudFront, and Highwinds.

8.2.3. Step 3. Empirical Performance Analysis

In order to assess the performance of CSP-hosted domains, we run a dedicated measurement campaign on the MONROE platform [79], the first open access measurement platform for independent and large-scale experimentation on commercial MNOs. MONROE consists of programmable nodes spread across several European countries, each one multi-homed to three MNOs. For this study, we use nodes in 4 countries — including SIMs performing international roaming — as listed in Table 8.1. We benefit from MONROE's openness and capabilities to run long-lived active measurements in realistic but controlled scenarios in commercial MNOs. Namely:

- **DNS test:** We run DNS lookups for each target mobile domain. We compare the response provided by the default MNO DNS resolver with Google's and OpenDNS public resolvers. This allows us to compare the quality of the responses and identify possible DNS-level inefficiencies during the replica selection. All of our DNS queries include the EDNS flag [69] as it may be used by CSPs like EdgeCast and Amazon services to locate the end user and improve the quality of replica selection [95]. If we do not receive a reply, as it occurs for Vodafone (IT) possibly due to the presence of an in-path DNS proxy filtering those requests, we repeat the request without the flag. We detect support for this functionality based on whether the DNS response includes an ECS option with the client's subnet [69].

- **Traffic performance test:** We measure the TCP connection and TLS session establishment time, if applicable, towards the resolved IP addresses. We open both TCP and TLS connections over TCP ports 80 and 443 by making an HTTP GET request for the `favicon.ico` object. The presence of the object in the server is not relevant as the handshakes are triggered regardless of its

Table 8.2: Top 5 FQDN by app penetration.

App (%)	SLD	IP(%)	CSP(s)
29	googlesyndication.com	0.8	Google
28	doubleclick.net	1.4	Google
27	facebook.com	0.6	Facebook, Akamai
26	crashlytics.com	1.4	AWS
25	googleadservices.com	0.5	Google

existence.

- **Network topology test:** For each resolved IP address, we run UDP traceroutes to study CSP-MNO peering and topological relationships.

We run the aforementioned experiments in isolation from other experiments continuously over 4 weeks, from April 5, 2017 until May 6, 2017. The combined results of the three tests for a given FQDN produce a "sample". The measurements are run continuously, and the time period between our samples varies between 4 and 24 hours. Thus, our experiments may cover several instances across time of day and various radio conditions. Nevertheless, we run the experiments sequentially to guarantee similar network conditions across FQDNs. We do not measure metrics such as Time to First Byte (TTFB) [96] and download speed as they are more likely to be affected by server-side artifacts, which is beyond the scope of this study. The measurement code and dataset are publicly available [94] to satisfy the reproducibility principle.

Data sanitation: We leverage metadata provided by the MONROE nodes to avoid bias introduced by uncontrolled changes in the wireless technology coverage (see Section 8.5 for further details) while we use packet captures to ensure that the measurements run by higher layer tools are accurate. We also remove measurements that may be affected by MNOs enforcing volume caps which may inflate latency. After sanitizing our dataset, we obtain a set of 173,679 valid samples.

8.3. CSP Prevalence on Mobile Apps and Services

In this section we analyse the prevalence of CSPs among mobile apps in order to identify the main players supporting the mobile Internet and multi-CDN strategies. 55 of the CSPs that we identified in the previous section are present in the Lumen dataset and 85.2% of the apps connect to at least one of them. However, such a high prevalence is not necessarily a consequence of app-developer decisions. As we can see in Table 8.2, advertising-related FQDNs have the highest app penetration [89, 97]. Just the domain `googlesyndication.com`, hosted by Google on its own datacenters, is present in over 29% of the apps in our dataset. Most organizations seem to leverage a single CSP. However, popular organizations like Facebook clearly follow a multi-CSP strategy by combining their own CSP infrastructure with Akamai's.

Figure 8.2 ranks CSPs by the number of mobile apps connecting to them, also showing the

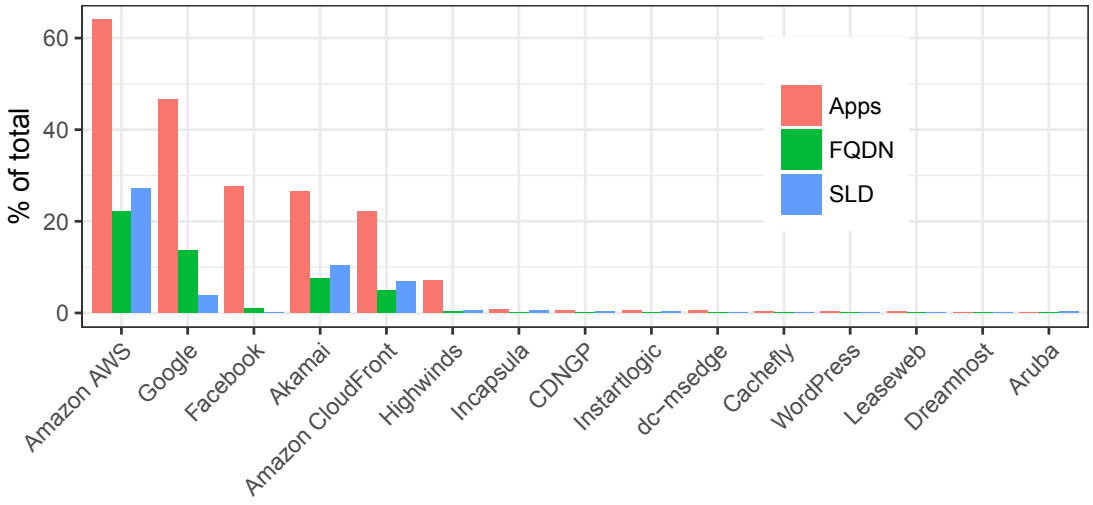


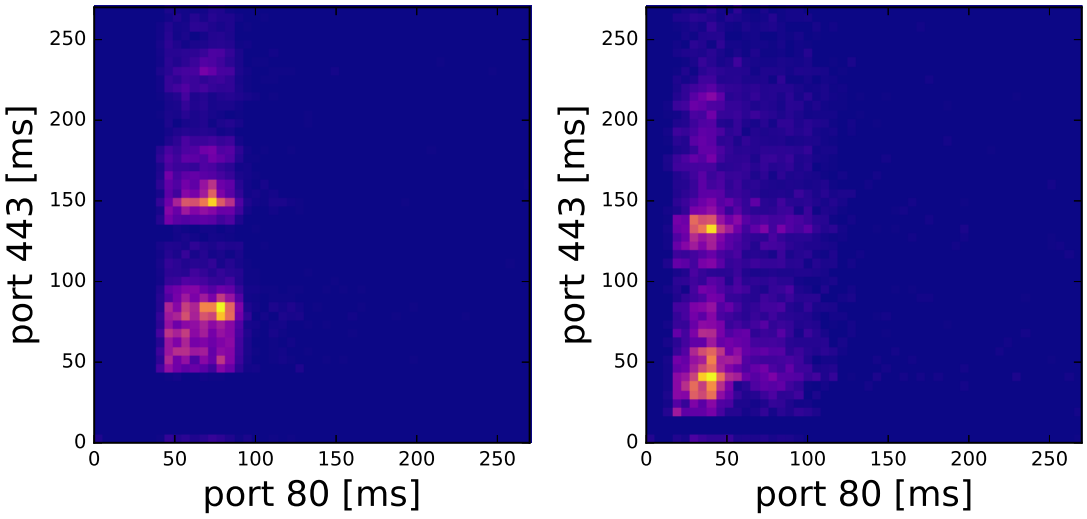
Figure 8.2: Top-15 CSPs prevalence by app, FQDN and SLD.

Table 8.3: Multi-CSP strategies by FQDN and SLD.

# of CSP	1	2	3	≥ 4
FQDN(%)	97.4	2.5	0.1	0.0
SLD(%)	84.9	13.6	1.3	0.2

percentage of FQDNs and second-level domains that they support. We report CSP prevalence by apps, by FQDNs and second-level domains (SLDs in short) to give a sense of both app and domain-level usage. The figure reveals a clear power-law distribution. While 49 CSPs (*e.g.*, Pure-peak, not shown in the figure) have a marginal presence as they receive connections from less than 1% of the apps, six CSPs play a central role on this market, being associated with 85.05% of the apps. AWS and CloudFront, both owned by Amazon, are the most used CSPs by mobile on-line services, supporting 27% and 6% of SLDs, respectively. Other CSP services like Facebook are easily found across mobile apps as many of them integrate Facebook services, including advertising, through the Facebook Graph API. However, Facebook also leverages Akamai's infrastructure as well as its own, which is only open to affiliate companies like Instagram. Google, instead, has opened their infrastructure to third parties with the Google App Engine service.

Finally, we leverage our PTR-CSP mapping to identify instances of multi-CSP strategies on a per-FQDN and per-SLD basis. According to our results (Table 8.3), only 3% of the analysed FQDNs and 15% of SLDs use at least 2 CSPs. After carefully inspecting such FQDNs, we can conclude that they are associated with large companies such as Samsung, Adobe, Chartboost, Unity or Facebook among others. This observation suggests that multi-CSP strategies are specific to large companies — probably because of cost-related reasons — despite their performance and reliability benefits.



(a) Location: Spain (roaming)

(b) Location: Italy

Figure 8.3: Heatmap of TCP connection times over ports 80 and 443 for a Vodafone Italy SIM when roaming (left) and when connecting from the home network (right). Lighter colors indicate more repetitions.

8.4. CSP Performance and Integration with MNOs

This section aims to analyse the actual performance of CSPs in the MONROE platform. To that end, we study first the presence of in-path middleboxes on MONROE's MNOs given that their presence can bias CSP performance measurements (Section 8.4.1). In Section 8.4.2, we characterize the DNS infrastructure deployed by our tested MNOs and their support for EDNS, a DNS extension used by many CSPs to correctly locate the end-user. Third, we study the CSPs' performance in the MONROE platform (Section 8.4.3) following the methodology described in Section 8.2.3, further analysing in detail the effect of topological and peering relationships between MONROE MNOs and our six target CSPs (Section 8.4.4) on TCP and TLS connection establishment. We conclude with an analysis of the impact of international roaming on TCP and TLS performance (Section 8.4.5).

8.4.1. In-path Middleboxes

MNOs may deploy Performance Enhancing Proxies (PEP) to optimize mobile traffic performance [62] using techniques like TCP splitting. However, TCP-splitting proxies can introduce bias in our measurements, as the TCP connection time obtained is to the proxy rather than to the final end-point (*i.e.*, the CSP). In order to identify such scenarios, we run the Netalyzr network troubleshooting tool [98] directly on the MONROE nodes [62, 99]. Netalyzr only revealed a TCP-splitting proxy for Vodafone Italy in TCP port 80. MNOs do not deploy TCP-splitting proxies on port 443 as their presence can interfere with mobile apps' securing TLS flows [100].

As a consequence, in the Spanish location, all the measurements over port 80 have a maximum connection time of about 130 ms, whereas over port 443, we cannot identify an upper limit as can be seen in Figure 8.3a. Likewise, in the Italian location, the vast majority of connection time measurements over port 80 are below 150 ms, whereas over port 443 we cannot identify an upper limit as can be observed in Figure 8.3b. The inflated times of the Spanish location are due to the roaming strategy of Vodafone Italy, as we will discuss in the sequel. Due to the above, we focus our TCP performance analysis only on TCP port 443, a middlebox-free path for all our MNOs.

8.4.2. DNS infrastructure

DNS Proxies: As for stateful TCP traffic, MNOs may also deploy DNS proxies to gain full control over user's traffic [62]. Their presence can interfere with CSP's performance by altering both DNS queries and responses. DNS resolvers deployed by all Swedish carriers perform cache delegation of DNS records (*i.e.*, they store the authoritative DNS servers for a specific domain), a common practice across MNOs [101]. Furthermore, TIM Italy proxies DNS traffic and performs DNS wildcarding (*i.e.*, they resolve non-existing names). Finally, Vodafone (IT) seems to actively block all DNS requests towards any DNS resolvers containing the EDNS flag. This behavior could be caused either by a DNS proxy or a DNS-aware firewall. Consequently, we cannot study the impact of the EDNS extension on Vodafone (IT).

EDNS support: Both Google's Public DNS and OpenDNS publicly claim to support the EDNS `edns-client-subnet` flag. We study whether DNS resolvers — including the default DNS resolver provided by the MNO as well as Google's and OpenDNS resolvers — support EDNS by checking whether the ECS option is included in the response, as specified by the RFC 7871 [69]. According to our results, only Google's public DNS seems to completely follow the RFC 7871 recommendations: the ECS option is absent from OpenDNS responses despite supporting EDNS [95]. This observation leads us to believe that OpenDNS may have their own interpretation of the standards. The only instance where the flag is absent from the response, when contacting Google is for TIM (IT), probably due to an in-path DNS proxy. Additionally, we check both the EDNS buffer size and reply size. For all the Norwegian operators as well as Telia SE, DNS replies are limited to 512 bytes, which is an indication that EDNS is not supported.

As opposed to public DNS resolvers, MNO's recursive DNS resolvers may not need to provide EDNS support to help the authoritative name server locate the end-user. MNOs may assign the same public IP address space both to end-users and recursive DNS resolvers. To study the feasibility of this, at least for the MNOs under consideration, we record the public IP of the user by sending traffic to a machine we own. Then, we get the public IP of the default DNS resolver of the operator, by resolving the URL `whoami.akamai.net`, which returns the IP from which Akamai servers receive the DNS request. Our experiment shows that all our studied MNOs use different address spaces for their DNS infrastructure and their subscribers, hence making the localization of the user based on the public IP address of the resolver impossible.

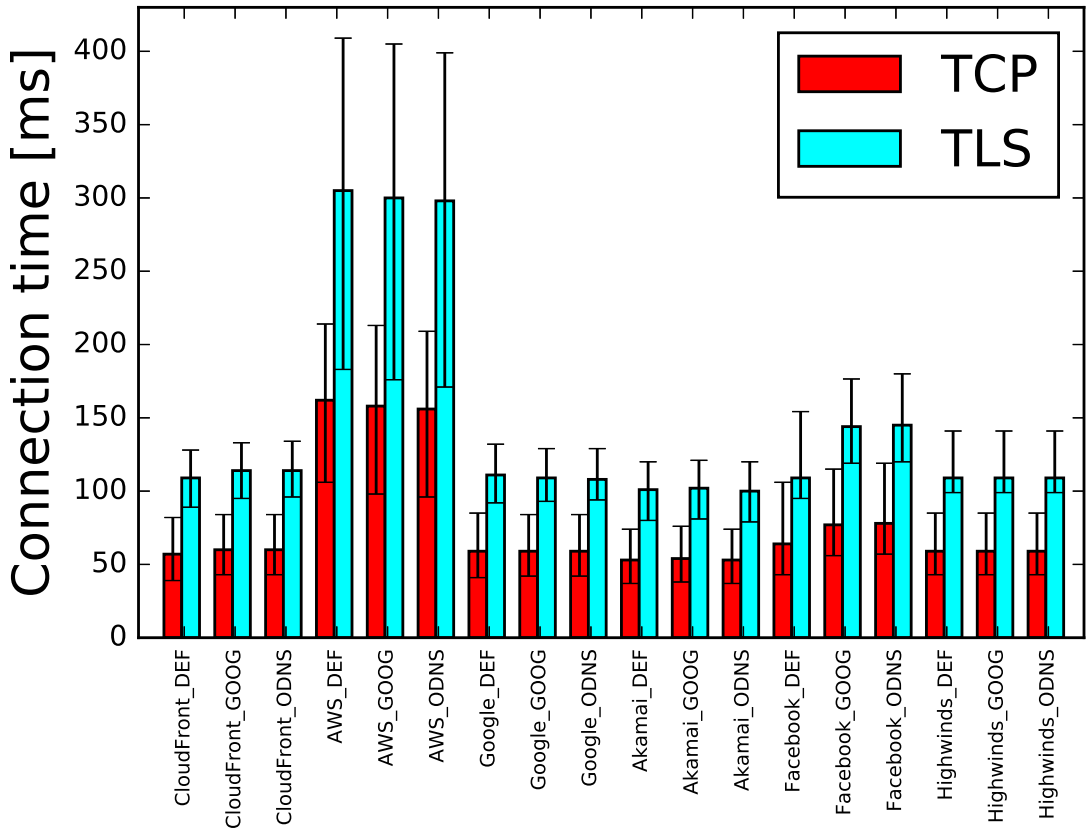


Figure 8.4: Median values of TCP connection time and TLS handshake duration for $<$ CSP $>$-$<$ DNS resolver $>$ combinations. Error bars represent the 25th and 75th percentile.

8.4.3. CSP Performance

Once we understand the DNS infrastructure of each MONROE MNO, we study the "quality" of the DNS responses provided by each DNS service and its impact on TCP and TLS connection time. In only 45% of the tests, the three resolvers return IPs that belong to the same set of \24 subnets. However, only 2% of the responses provided by the third-party DNS providers point to machines hosted in a different country than the one in the responses of the default MNO resolver. In that case, the increase of the median handshake duration is at most 2ms and 9ms for TCP and TLS, respectively.

Figure 8.4 shows the TCP and TLS connection time across all our MNOs, grouped by the DNS resolver being used. As we can see, the three resolvers perform similarly at the transport level regardless of EDNS support. MONROE nodes connecting to IPs provided by the default DNS resolver have marginally better TCP connection times than those connecting to servers proposed by OpenDNS and Google's public DNS (around 3ms). Facebook is the only CSP exhibiting significantly better performance when using the MNO's default resolver in all MNOs but in the case of TIM (IT) where all the DNS responses perform similarly because of its in-path DNS

proxy. For the other MNOs, the responses provided by the default DNS resolver are 10ms faster at the TCP level compared to Google's and OpenDNS' responses.

Figure 8.4 also reveals performance differences across CSP services, namely due to the scale and coverage of their infrastructure. As we can see, Amazon CloudFront (a CDN provider with a vast infrastructure) and Amazon AWS performance differences are remarkable. A possible explanation for that is that CloudFront is a dedicated CDN provider service, with replicas in multiple locations, whereas AWS is a cloud computing platform with just a few data centers in Europe. In fact, among the many FQDNs studied, we can identify that many Amazon AWS customers do not leverage AWS's global infrastructure and decide to host their services entirely in US data centers, hence further inflating the delays.

8.4.4. CSP-MNO Integration

In this section, we study the topological relationships (*e.g.*, peering) and the geographical distribution of CSPs' data centers and MNOs' Points of Presence (PoP) in order to identify their effect on TCP and TLS connection time. We use MONROE's traceroute measurements to characterize the interconnection between MNOs and CSPs. We retrieve the Autonomous Systems (ASes) that advertise in BGP the most specific network prefix covering the IP of each hop in the traces we collect.

Given the difficulty to accurately measure the geographic distance between an MNO and a CSP without insider knowledge, we define the following distance metrics:

- **Country distance:** This metric counts the number of unique countries traversed by a traceroute probe from the MONROE vantage point to the target CSP. We retrieve country-level information by mapping each hop's IP along the data path to a country code using MaxMind's free GeoIP service. Consequently, MaxMind's accuracy [102] constrains our analysis accuracy.

- **Organization distance:** This metric reflects the number of unique organizations traversed by a traceroute probe from the MONROE vantage point to the target domain. For each hop's IP address along the traceroute data path, we retrieve the AS using the most specific prefix advertised in BGP. Then, we use CAIDA's AS-to-Organization mapping dataset [103] to identify the parent organization for each IP address[2].

Large Internet entities such as Tier-1 ISPs and CSPs may use multiple AS numbers and yet advertise various IP blocks with the same origin AS [104]. This makes identifying the actual entity behind a given IP block and its usage harder. For example, Telia Sonera – an ISP offering both fixed and mobile connectivity that owns multiple AS numbers – advertises its reachability information in BGP, including its mobile subscribers, using the AS number associated to its Tier-1 ISP.

[2]The same organization may own several AS numbers, thus we refrain from using the AS path length as a distance metric.

Table 8.4: Median and standard deviation values of the organization and country distance per CSP when aggregating all the MNOs that we measure in the MONROE platform. We target six main CSPs we previously identified in the analysis of the Lumen dataset.

CSP	Performance Tests #	Country dist. median (std)	Org. dist. median (std)
CloudFront	408,537	3 (1.6)	2 (1.0)
AWS	400,240	2 (1.3)	2 (1.0)
Google	69,839	2 (0.8)	2 (0.7)
Akamai	67,375	2 (0.7)	2 (0.7)
Facebook	9,731	3 (1.1)	3 (0.9)
Highwinds	8,984	3 (0.9)	3 (0.7)

Table 8.5: The effect of organization and country distance on TCP connection time [median (std)].

CSP	MNO	Organization dist.	Country dist.	TCP conn. time [ms]
Akamai	Telia NO	2 (0.3)	2 (0.4)	42.0
	Telenor NO	3 (0.9)	2 (1.0)	65.0
	Telia SE	3 (0.6)	2 (0.4)	52.0
	TIM IT	2 (0.3)	2 (0.5)	30.0
	Orange ES	2 (0.5)	2 (0.7)	39.0
	Yoigo ES	2 (0.7)	2 (0.5)	38.0
Google	Telia NO	3 (0.0)	2 (0.0)	43.0
	Telenor NO	2 (0.4)	2 (0.4)	75.0
	Telia SE	3 (0.7)	2 (0.3)	57.0
	TIM IT	2 (0.5)	2 (0.5)	30.0
	Orange ES	2 (0.4)	3 (1.2)	56.0
	Yoigo ES	2 (0.4)	3 (0.8)	41.0

Results: Table 8.4 presents the median country and organization distance for each CSP across all our MONROE nodes. We identify organizations and countries along the path based on the IPs we see in the traceroute data. In general terms, Akamai and Google have the smallest distance metrics, presumably because of the extensive use of peering and the presence of caches within MNO networks. Table 8.5 shows the impact of distance values on the TCP connection time for a selection of MNOs and CSPs. When aggregating all results across all MNOs, we can see that most flows cross 2 or 3 organizations on average at most. The following paragraphs discuss specific MNO cases.

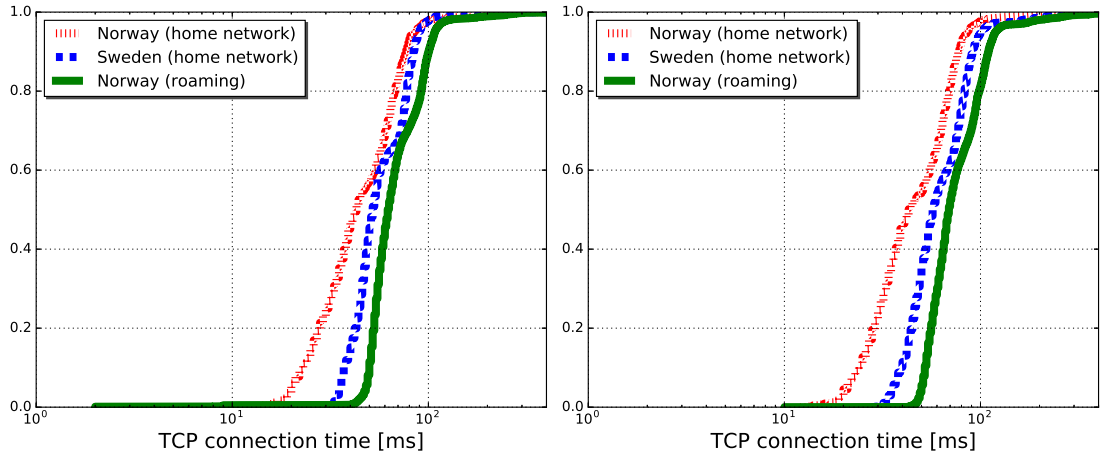
Telia (SE, NO): Both Telia and Telenor operate in Sweden and Norway. The median value organization distance between Telia and the majority of our target CSPs is 3, regardless of the operating country. The only exception is Akamai for Telia NO, where the median value goes down to 2 organizations. From our MONROE measurements, we find that Telia (SE) (AS3301) reaches over 85% of the Akamai services directly through its Swedish parent organization, Telia Company (AS1299), which also acts as its Internet transit provider [105], hence possibly inflating

the network path. Similarly, we find that Telia (NO) (AS12929) always routes its traffic through Telia Company (AS1299). which has over 1,500 customer networks and 50 peers. The ongoing strategic alliance between Akamai and Telia [106] allows the CDN-MNO collaboration to improve the end-user experience in a cost-effective manner. We measure a median TCP connection time of 52ms and of 42ms for Telia SE and Telia NO to Akamai, respectively. According to [105], Telia Company registers as a peer for Google Inc. (AS15169), explaining the organization distance of 3 to Google from Telia NO and Telia SE. The TCP connection time we measure towards Google servers is similar to Akamai's prior values: 43ms from Telia NO and 57ms from Telia SE, respectively.

Telenor (SE, NO): Telenor Sweden's median distance towards each one of the six CSPs is 3, while for Telenor NO the distance varies in median value with Google having the smallest value (2 organizations) and Akamai – a median value of 3 organizations. Telenor NO (AS8786) belongs to the Telenor group and is registered for mobile operations in Norway. AS8786 always routes its traffic using the parent company AS2119. Similarly, Telenor SE also depends on the same AS2119 to reach targets. Our traceroute measurements reveal a high diversity of AS-level paths when reaching Akamai target servers with a median organization distance of 3. In 20% of these paths, we observe IP address blocks belonging to the Amsterdam Internet Exchange (AMS-IX). This suggests that Telenor (NO) leverages its peering connections in the Netherlands at AMS-IX to reach content hosted by Akamai. This result does not suggest an ongoing alliance between Telenor and Akamai. This observation translates in performance degradation: 65ms in median value from Telenor NO to Akamai (Table 8.5) In Telenor SE we also identify that 27% of the requests have a country distance of 4 or higher, while for the rest of the MNOs this is less than 2%.

TIM (IT): Telecom Italia (AS6762) (TIM) is one of the largest operators in the world, with peering connections to all the other Tier-1 ISPs [105]. Thanks to its dense global interconnection and large user-base, CSPs such as Akamai entered into partnerships agreements with TIM to optimize content delivery and increase the quality of experience of content consumers [107]. We note that when breaking down the organization distance (Table 8.4) for TIM on the six different CSPs, only Facebook and Highwinds have a median distance value of 3 organizations, while for the rest we find a median distance value of 2 organization. The tight integration of TIM with Akamai and Google translates into low TCP connection times: 30 ms both for Akamai and Google.

Yoigo (ES) and Orange (ES): The two MNOs that we measure in Spain, Yoigo and Orange, present similar distances to the two CSPs listed in Table 8.5. Our traceroute experiments show that Yoigo (AS16299) relies exclusively on its two transit providers to reach popular content, namely Telia Company (AS1299) and Orange Spain (AS12715). Similarly, Orange ES relies on its two providers, Orange S.A. (AS5511) and Level 3 (AS3356) to reach both Akamai and Google services. Both MNOs have Tier-1 ISPs as providers, and leverage the latter's dense interconnection with Google and Akamai to ensure good performance for their customers. Our measurements re-



(a) Akamai (b) Google

Figure 8.5: Effect of roaming on TCP handshake over Telia.

port a median TCP connection time of 39ms from Orange ES to Akamai and of 38ms from Yoigo ES to Akamai. The TCP connection time from both MNOs is slightly higher towards Google (Table 8.5), which may be caused by a higher median country distance.

8.4.5. International Roaming

When a SIM card is in international roaming state, MNOs can either forward their traffic to the home network before reaching the Internet (*i.e.*, home routing) or use the host MNO infrastructure (*i.e.*, local breakout) [62]. Most MNOs decide to implement home routing so that they can keep control over their subscribers' traffic at the expense of inflated path length and as a consequence downgraded performance.

Our traceroute analysis reveals that our MNOs implement home routing roaming as the number of hops to a given target remains the same – due to the presence of a transparent tunnel – but path latency increases. Figure 8.5 presents the CDF of TCP connection time of all the successful connections over ports 80 and 443 in logarithmic scale for Telia SIM cards. We compare the performance for the three Telia SIM cards in our MONROE nodes: two locally connected in Norway and Sweden, and a third Telia Sweden SIM card roaming on a Norwegian MONROE node. According to our results, TCP connection time takes on average 20ms more on roaming devices than on those connecting directly to the local network (Figure 8.3) because of the country distance inflation. The home routing roaming approach, by its nature, defeats the purpose of CDNs placing content close to the user. Roaming users do not benefit from existing peering agreements between the host network and large CSPs. For this reason, when the target server belongs to a well-provisioned CSP— possibly better peered — this impact is greater, compared to CSPs present in a few locations. For example, CloudFront and Google services over the network of Telia have at least 20% delay inflation. In contrast, the delay inflation for AWS under the

same conditions is usually below 15%. We also identify a clear performance degradation in the Vodafone (IT) SIM card roaming in the Spanish node. It is visible in Figure 8.3b, where the point cloud is located further away from the axis (higher delay) compared to the local connection.

8.5. Study Limitations

App representativity: Our mobile app sample is limited to the traffic logs obtained from our Lumen users. Nevertheless, as discussed in our previous work [89], we consider the apps in our dataset to be representative of those used by average mobile users from all over the world: 48% of the apps in our records have more than 1M installs while 71% of the apps listed on the Google Play Top-50 charts for USA, Spain, Germany, India and UK are also present in our dataset.

CSP representativity: Because of limited testing cycles on the MONROE platform, our study focuses on the six most representative CSPs across our sample of mobile apps. However, these CSPs are likely the better peered ones with large MNOs due to their popularity. Finally, we intentionally execute measurements towards FQDNs rather than towards specific CSPs to analyse realistic domains and characterize DNS-level artifacts. Unfortunately, this is skewing our number of samples towards the most popular CSPs.

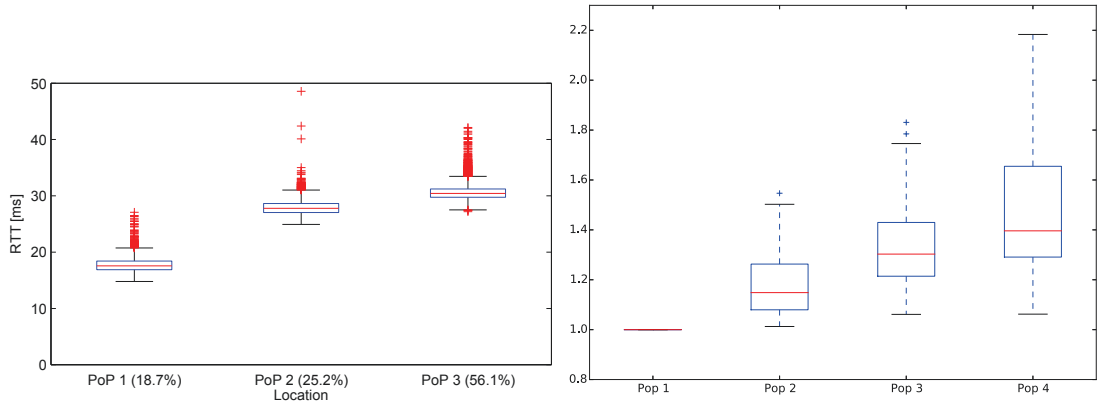
Cellular technology: We only consider measurements run only over LTE due to its rapid adoption rate and its low latency radio link. Including 3G and 2G cellular technologies in our studies could bias our empirical results due to the significant differences in the radio access link.

Active measurements: We execute our active measurements in real networks against real systems. As a result, the scale and accuracy of our results may be limited by a number of factors beyond the scope of this study. We cannot fully control aspects such as CSP load balancing mechanisms and server load, cellular network behavior, network load and congestion, and radio link stability which may influence and introduce bias in our results.

Given the aforementioned limitations, our goal is not drawing conclusive causal relationships between MNOs and CSPs, but providing a first study of this complex ecosystem to motivate further research. To that end, we made public our data and measurements scripts so that other researchers can continue, extend and improve our work.

8.6. Effect of Point of Presence (PoP) Selection on Quality of Service (QoS)

We conclude this chapter by presenting a relevant measurement study we performed a few years prior the campaign analysed in the previous Sections. The part of the study we will discuss in this thesis, reveals that the choice of PoP may increase end to end delay by more than 50%, while it is very frequent to be assigned a suboptimal PoP. Therefore, this section emphasizes the importance of proper operator configuration when accessing third-party services.



(a) Effect on minimum RTT towards a measurement server across the detected PoPs.

(b) Effect of suboptimal PoP selection on the median TTFB of Germany's 25 most popular domains.

Figure 8.6: Boxplots presenting the effect of suboptimal PoP selection in two scenarios (Figure from [2]).

The main dataset we discuss was collected in the city of Darmstadt, Germany between 19 December 2014 and 19 January 2015. We specifically select this period in order to include samples during normal work weeks and holidays. In order to minimize the effects of weak hardware, we use five state of the art at the time, Nexus 5 devices. We measure the long term variation of:

- RTT towards a well connected dedicated server located at TU Darmstadt.
- TTFB of the, at the time, 25 most popular German websites, based on the Alexa ranking [108]. The measurement retrieves only the main HTML document.

All the RTT measurements use the native Linux `ping` and `traceroute` command line utilities to avoid the overhead added by the Android API [58]. Each ping consists of five probes and we ensure that the device is in "RRC connected" state before each measurement to avoid including promotion delays in the dataset. When measuring the RTT of the different hops detected by traceroute, we only keep the minimum value of three probes, to counter potential routing delays caused by the low priority of the ICMP unreachable messages. In order to reveal as many hops as possible, we complement the phone `traceroute` measurements with traceroutes from the measurement server towards the public IP of the phones. We then combine the two traceroute logs creating a more complete view of the path. We perform a `ping` measurement every minute and a `traceroute` measurement every 15 minutes.

When measuring website performance we rely on TTFB in order to be certain we are measuring the performance of the host server and not of a PEP. This set of measurements has a periodicity of 30 minutes. We rely on the operator's DNS server to resolve the URLs, expecting to be directed to the best performing server. Finally, we only take into account measurements performed using LTE technology.

The ping and TTFB measurements reveal delay, while the traceroute measurements locate the PoPs. We are able to distinguish PoPs based on the host names of the routers close to the edge of the operator's network. First, we discuss the dataset of the measurements towards our own server and present a summary of the results in Figure 8.6a. 3 PoPs are identified in these measurements. We group measurements per PoP and study the minimum RTT values of the pings towards our server. The distributions of the minimum RTTs per PoP are presented as boxplots. In the parenthesis of the x-axis we present the relative frequency of each PoP in the dataset. The best performing PoP was used the least, serving only 18.7% of the measurements, whereas the worst performing PoP had the majority of the samples with 56.1%. Each PoP has a relative constant performance over time, as denoted by the small size of the boxes, symbolizing the 25%th and 75% percentiles. The end–to–end delay increase, compared to PoP 1, is 58% for PoP 2 and 73% for PoP 3.

The reason for the vast performance difference is the location of phones, server and PoP. Both the phones and the server are located at the same city. Thus, when the traffic is routed through a PoP of a nearby city, the overall path is short. On other cases though, the PoP is located at the other side of Germany. As a consequence, the traffic has to travel towards the PoP moving away from the server and then move back towards the server through the public Internet, inflating the delay unnecessarily. This comes in contrast to the common assumption that traffic is routed through the shortest path. Rather, it is distributed across the operator's network. We conclude that the internal routing of the operator has a very big impact on the performance of a well connected server and since the majority of the traffic is routed suboptimally, there is a lot of room for improvement.

The above has obvious implications on CDN performance. We quantify the impact on CDNs by measuring the most popular German websites, which are highly likely to use CDN services. As before, we group the measurements per PoP. We are able to detect again the previous 3 PoPs, as well as a new one not used before. To quantify the performance difference, we calculate the median TTFB of each URL–PoP combination. A URL–PoP combination is essentially a different path between the same endpoints. Considering as a baseline the performance of each URL over PoP 1, we calculate the relative difference of the URL performance over the rest of the PoPs. The distribution of the relative increase in delay for each of the suboptimal PoPs is presented as a boxplot in Figure 8.6b. We observe that the median value of these distributions is between 20% and 40% higher for the suboptimal PoPs compared to PoP 1. We further statistically validate the significance of the performance impact of PoP selection on the delay, by running a Kruskal–Wallis test on the sample. Therefore, PoP location should be taken into account, alongside server location and user location, when matching users to CDN servers.

Next, we try to examine the criteria of the PoP selection. We can only observe that the PoP changes in specific time periods, which are multiples of 36 hours. As consequence, the performance over time has the same periodicity as the change of PoP. Figure 8.7 reveals this pattern. It presents an histogram of the PoP and public IP assignment durations, with a bin size of 6 hours. The selection of PoP appears to be random and can be forcefully triggered before

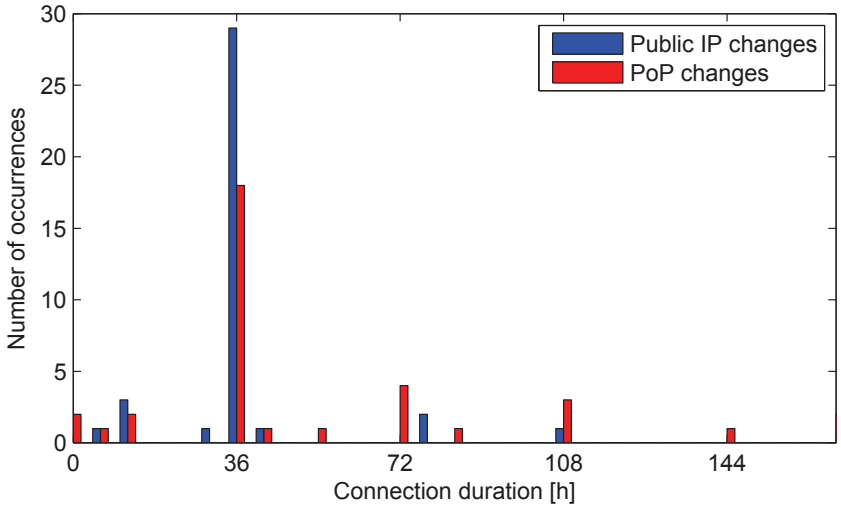


Figure 8.7: Histogram presenting the duration of external IP leases and PoP selections (Figure from [2]).

the timeout period by disconnecting and reconnecting the device to the network. The selection of PoP is also independent of the public IP address. The public IP address, which changes more frequently than the PoP, does not seem to have any effect on performance.

We also observe that, for the whole duration of the study, all devices were connecting to the same DNS server address. Alongside every traceroute, we performed an RTT measurement towards the DNS server. We measure smaller delays contacting the DNS server, while connected to a better performing PoP, indicating that DNS Servers are located close to the PoPs. Thus, PoP choice also affects the trip duration towards the DNS server.

Finally, we conduct two smaller measurement campaigns in Denmark and Spain, using different devices and network technologies, such as 3G. The public IP changes with a periodicity ranging between 3 and 24 hours in the Danish network and exactly every 12 hours in the Spanish one. We are able to detect a single PoP in the Spanish network with very little variation in performance.

8.7. Summary

We have presented an overview of the interconnection and performance of MNOs and CSPs. We leveraged accurate traffic fingerprints from thousands of mobile apps that we collected through crowd-sourcing with Lumen [78]. This data allowed us to i) identify the most relevant CDNs and cloud providers for mobile traffic; ii) map their connectivity with relevant European MNOs; and iii) measure their performance using the MONROE platform [79]. Our results show a significant reliance of apps on mobile CSPs with the major CSPs being used by 85% of the apps. We reported path inflation (*e.g.*, due to poor peering relationships and roaming) and presence of middle-boxes (*e.g.*, in-path DNS proxies) which can significantly impede CSP performance, but we saw no

noticeable difference in performance metrics when using different DNS resolvers or enabling the EDNS parameters. Our active measurement dataset, the code for the measurement experiments and the CSP mapping tool are publicly available [94]. We have further shown that the choice of PoP may have a significant impact on the performance of CDNs.

Chapter 9

Conclusions

In this thesis, we have proven the feasibility of performing accurate smartphone measurements, presented techniques to passively measure mobile bandwidth by observing minimal traffic and identified inefficiencies in the interconnection of mobile operators with CSPs. These may be used to optimize the delivery of multimedia to mobile devices without investing in expensive infrastructure or requiring big modifications in existing components. We also expect that the accuracy of our tools will improve thanks to the rapid increase in the computing power of mobile devices and recent improvements in the networking stack of the Linux kernel.

We started with giving a detailed presentation of how a packet travels from a content server to the smartphone application requesting its payload in Chapter 2. This chapter serves as a foundation of the work presented in Part II. To my knowledge, this is the only public resource that presents all the steps accurately and cohesively. Then we presented the State of the Art in the areas of bandwidth estimation and measuring CSP performance in Chapter 3. In contrast to past research, this is the first study proposing solutions to estimate the bandwidth of mobile devices passively, while requiring very little traffic. We are also the first to identify and assess the relationship of CSPs and MNOs in the European ecosystem.

Part II starts with Chapter 4 assessing the feasibility of performing accurate measurements on smartphones. We proved that LTE data rate measurements performed by mobile phones can be accurate and precise, but depending on the acceptable margin of error a calibration of the specific devices might be necessary. The measured rate does not affect the relative error which is somewhat constant per phone. Since the kernel is closer to the physical layer, measurements at this level may have better error compared to the application level. Due to the the mechanics of LTE scheduling downlink estimates are better than uplink ones. The bigger duration of packet bursts is key to generating better estimates than packet groups, but in the following chapters we developed algorithms that may use packet groups in challenging scenarios to achieve good results. Chapter 5 went in depth in identifying the route causes of measurement artifacts on the kernel level. It also explained why the state of the art lightweight bandwidth estimation techniques are not applicable to mobile networks.

The last 2 Chapters of Part II build upon the analysis of Chapters 2, 4 and 5. Chapter 6 presented a very lightweight mobile bandwidth estimation tool that is able to provide reliable results by monitoring small data exchanges. To the best of our knowledge, it is the only tool that may provide an estimation relying solely on the traffic generated during the early phase of a TCP connection in mobile scenarios. Such flows are the vast majority in mobile networks. In a streaming scenario, they may provide enough information to choose the ideal initial streaming quality the bandwidth of the consumer may support. Chapter 7 adapted traditional packet dispersion techniques to mobile networks. This allowed for generating accurate per user capacity estimations, by exploiting as few as 5 % of the information obtained from TCP data flows. Given that this solution can support dense throughput sampling, it is ideal for capacity prediction and optimized resource allocation. If the future capacity availability is known, it is possible to predict when it is best to communicate by doing so when it is cheaper (*i.e.*, more capacity available), as will be presented in the following Appendix A. In addition, our solution is able to estimate the fast capacity variations from a mobile terminal by monitoring the traffic generated under normal daily usage.

Part III changes scope and focuses on identifying potential inefficiencies at the network side of the communication path. In Chapter 8, we performed the first holistic analysis of the complex ecosystem formed by mobile applications, CSPs and MNOs. We aimed to comprehensively characterize their relationships and dynamics and measured their performance with dedicated active measurements. We identified the dominant players and their prevalence in mobile applications with data collected through crowd-sourcing with Lumen Privacy Monitor. Using the MONROE mobile broadband testbed, we were able to study their interconnections in Europe and asses their performance and how it is affected by recent developments such as free roaming and extensions of the DNS protocol. We also showed that operators' configurations, such as the choice of PoP may severely impact their performance.

The work presented in the previous chapters and the following appendix has been published in six conference and two journal papers. At the time of writing these lines, we are awaiting the publication of the extension of our work on the accuracy of smartphone measurements presented here in Chapter 4 and published in [1], as well as working on extending the work on the interconnetion of CSPs with mobile ISPs presented here in the first Sections of Chapter 8 and published in [27]. Some of the tasks we have planned in the future include performing CSP related experiments in nodes installed in countries outside of Europe like USA and South Korea, as well as in mobile nodes (*i.e.*, trains that crosss a country).

To conclude, this work has proposed a series of improvements and identified inefficiencies in the current state of mobile networks. As the users' expectations continue to rapidly increase and the industry moves towards the new generation of mobile technologies, collectively called 5G, we can expect denser deployments, content placement even closer to the user, more powerful smartphones and smarter allocation of resources. The proposed improvements will have an even greater impact in this new environment, where a few ms of delay accounts for a big percentage

of the total roundtrip. Further, frequent data rate updates from mobile devices can enable base stations to perform smarter scheduling. Finally, we believe our work demonstrated that better tracking of packet arrivals is worth being adopted in future mobile OS releases.

Appendices

Appendix A

A Model for Throughput Prediction for Mobile Users

In this appendix, we propose a stochastic model to predict user throughput in mobile networks. In particular, the model accounts for uncertainty such as random phenomena (e.g., fast fading) or inexact information (e.g., user location) to derive the statistical distribution of the user throughput. Such a model is highly useful for aiding scheduling and resource allocation decisions. For example, the techniques mentioned in Chapters 6 and 7 can be used to feed the model with data and then obtain future values with the same granularity as the granularity of the input data. These future values may then be fed to a scheduling optimization algorithm like the one proposed in [10].

In addition, we provide a taxonomy of prediction techniques to investigate error sources and the main characteristics of prediction accuracy. Finally, we show the versatility of the model by analysing LTE user throughput for the case where knowledge of either the user's actual position or the congestion level in the cell is inexact.

The main contribution of this appendix is a novel synthetic model representing the impact of estimation and prediction errors on the bandwidth availability statistics to be able to study network resource optimization problems under forecasting uncertainties. In order for the model to account for the many different error sources, we analyse state of the art prediction models for both network resources as well as user mobility, which we subsequently organize in a taxonomy based on the time-scale and granularity of the prediction.

The rest of the appendix is structured as follows. Section A.1 provides an overview and taxonomy of predictors upon which our model is based. In Section A.2, we discuss in detail the model for network resource availability under estimation and prediction errors. The model is applied to LTE cellular systems in Section A.3 and Section A.4 concludes the appendix.

Table A.1: Prediction Taxonomy

Ref.	Cat.	Accuracy	Notes
[109]	(1)-net	$c_r \sim 0.8$	Provides a model for the number of user in a cell.
[110]		$\varepsilon \sim 0.15$	ARIMA models and wavelet MRA.
[12]		$\varepsilon \geq 0.01$	GARCH-ARIMA accurately models static high-speed network traffic.
[111]		$\varepsilon \in [0.01 - 1]$	Evaluates multi scale and s-sample prediction.
[112]	(2)-cell	$c_r \in [0.5 - 0.72]$	Compares Markovian (better) and Lempel-Ziv models.
[113]		$\varepsilon_l \sim 2$ m	User trajectory prediction.
[114]		$c_r \in [0.2 - 0.7]$	Route prediction on GPS data.
[115]		$c_r > 0.8$	Using pre-filtered data and Markov models. Prediction possible in the 98% of the cases.
[11]	(3)-user	$\varepsilon \in [0.05 - 2]$	Empirical study on user traces using wavelet approximations and filtering.
[116]		$\varepsilon \sim 1$	First attempt at mobile system bandwidth prediction.
[117]		n/a	Complete solution for mobile bandwidth forecast.
[118]		$\varepsilon \sim 0.01$	Spatial and temporal dynamics characterization of mobile Internet traffic.

A.1. Taxonomy of Predictors

In this section, we analyse predictors for both user mobility (A.1.1) and network resource availability (A.1.2) in order to understand the forecasting capability for mobile systems and the accuracy of the available solutions. The considered works cover a wide range of time scales, location granularities and levels of accuracy. To provide a comprehensive model, we classify them in three categories according to their time and space granularity.

The first group [12, 109–111], (1)-*net*, is the most coarse: network performance is modeled by analysing the whole network at once, with a time scale on the order of minutes to hours; users are statistically mapped to base station cell ID or geographic location, i.e., predictions obtained by these models concern average throughput achievable in the location a given user is most likely to be found.

Algorithms in the second group [112–115], (2)-*cell*, combine user mobility information and network location specific information to refine prediction granularity. Predictors belonging to this group aim at predicting the next cell a user is likely to visit, the congestion level in that cell and the time of the visit. Its timescale is between tens of seconds and a few minutes.

The third group [11, 116–118], (3)-*user*, comprises the predictors with highest time granularity: in fact, most of the solutions in this group leverage filtering techniques and historical data. The aim, here, is to model the fast bandwidth variations experienced by the users on a timescale of tens of milliseconds up to a few seconds.

Table A.1 groups the papers into the three categories and also provides a high level description of the papers. The "Cat." column specifies the name of the category, while the "Accuracy" column provides an evaluation of the effectiveness of the techniques. Here, we use the ratio between the

mean square error of the prediction and the standard deviation of the original time series (e.g.: the user throughput, the bandwidth availability, etc.) $\varepsilon = \mathrm{MSE}(\tilde{x})/\sigma_x^2 = \sum_i (x_i - \tilde{x}_i)^2 / \sum_i (x_i - \mu_x)^2$, where x_i and $\tilde{x}_i$ are the i-th samples of the original time series and their predictions, respectively, and σ_x and μ_x are the standard deviation and the average of the original time series, respectively; c_r is the correct prediction rate defined as the ratio between the number of times the predicted location of a user is correct and the number of attempts; and ε_l represents the distance between the predicted and the correct user position.

A.1.1. Mobility Predictors

The most common methods to locate a mobile terminal are, in order of decreasing accuracy, the Global Positioning System (GPS), WiFi, and cellular network positioning. These solutions can identify a terminal's position with an average error on the order of 10, 100 and 500 meters, respectively [119].

Theoretical works, such as [120] and [121] studied characteristics of human behavior and found that an appreciable level of self-similarity exists among behavioral patterns and that, within due limits, forecasting is possible. Among the many studied properties, we highlight the one asserting that the probability of a user to be found in a given location is approximately inversely proportional to the location rank.

Some predictors aim at estimating the next user position on a grid representing network cells: $[112]_{(2)}$ compares Markovian and Lempel-Ziv models trained with the sequences of locations a user visited in the past, while $[109]_{(1)}$ studies the accuracy of mobility modeling. Notably, the first paper comes to the conclusion that second order Markov models provide a good trade off between complexity and accuracy achieving a correct prediction rate $c_r \in [0.5-0.72]$ on mobility traces collected from more than 6000 users of Dartmouth College's wireless network. The second paper provides an effective way to estimate the number of users in a cell and, consequently, the congestion level.

Other predictors deal with routes and trajectories. $[113]_{(2)}$ uses 1-sample predictions of user position to improve the performance of a routing protocol. (An s-sample prediction computes the first s unknown samples of a given time series.) The location prediction accuracy is claimed to be on the order of a few meters, with a position error $\varepsilon_l \sim 2$ m. The work in $[114]_{(2)}$ focuses on predicting complete routes from historical GPS data and obtains a $c_r \in [0.2 - 0.7]$. Here, the best results are obtained when excluding single trips from the dataset.

Finally, $[115]_{(2)}$ uses second and third order Markov models trained on a pre-filtered leap graph to model and predict cellular user mobility. The solution is able to achieve a $c_r \geq 0.8$ in 98% of the cases. Finally, recent works, such as [122] and $[117]_{(3)}$, directly exploit position information obtained from navigation systems to map bandwidth availability to locations. While these solutions provide a prediction that is based on the actual intended destination of the user, the accuracy of the prediction is still limited by the accuracy of the positioning system and the possibility of user detours. To the best of our knowledge, a detailed study linking location prediction

accuracy to bandwidth/throughput prediction accuracy does not exist.

A.1.2. Bandwidth Predictors

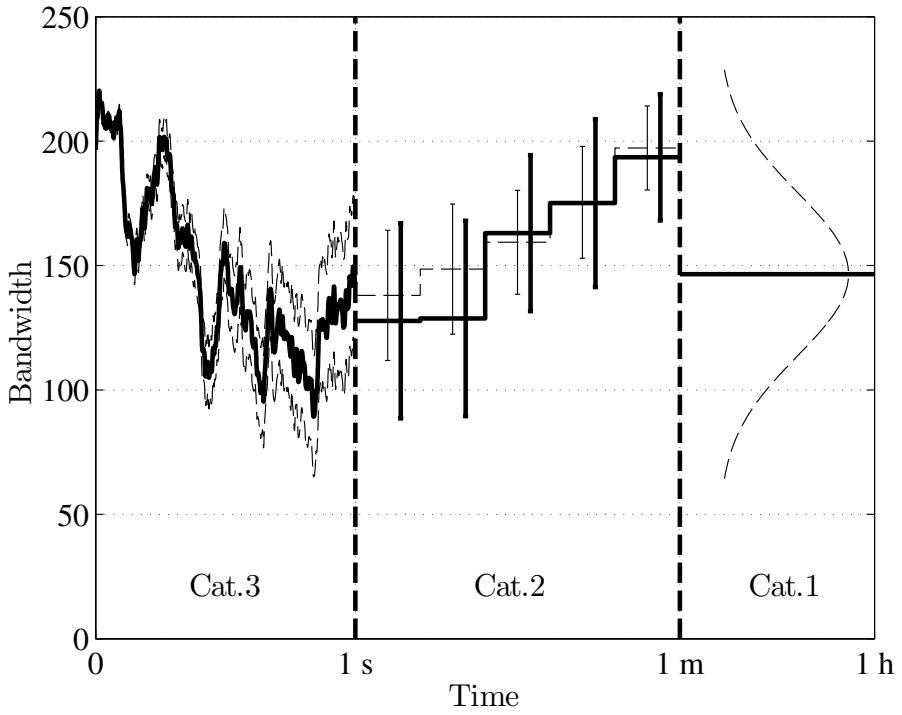


Figure A.1: Bandwidth forecasting examples: category 3, 2 and 1 predictor outputs are shown on the left hand side, in the center and on the right hand side, respectively.

One of the most relevant studies on traffic dynamics for cellular networks is [123], which conducted the first detailed wide scale analysis of network usage and subscriber behavior. The paper characterizes mobility and temporal activity patterns and identifies their relation to traffic volume. Traffic has been analysed from the base station point of view, identifying its variations over space and time.

Earlier works such as $[110]_{(1)}$, $[11]_{(3)}$ and $[12]_{(1)}$ studied different filtering techniques, namely MEAN, LAST, BM, MA, AR, ARMA, ARIMA, and FARIMA, all of which are different combinations of moving average and autoregressive filtering. We refer the interested reader to the source papers for the details.

Although different papers use slightly different metrics, the following conclusions can be drawn: low order filtering techniques coupled with smoothing solutions (e.g., wavelet MultiResolution Analysis (MRA) or wavelet approximation) are able to provide 1-sample static network traffic predictions with an error as low as $\varepsilon = 0.05$ and almost always lower than the variance of the original signal, $\varepsilon = 1$. (In the latter case, the predicted sample error would be as large as those that would have been obtained by generating random samples from a distribution with the same variation as the original signal). The error decreases with larger timescale and smoother

approximation of the signal.

Subsequent work in $[111]_{(1)}$ compares FARIMA and GARCH filtering techniques in terms of both time scale and the number of predicted samples s. Results obtained from Internet traffic traces show that GARCH outperforms FARIMA, achieving an error that is four times smaller. The authors confirm that the error decreases with increased signal timescales and increases with the number of predicted samples s. In particular, the error becomes as high as the variance of the original signal for $s = 10$ and $s = 100$ samples for FARIMA and GARCH, respectively. Also, GARCH errors are slightly smaller than half the variance for $s = 10$ samples and beyond.

$[116]_{(3)}$ and $[118]_{(3)}$ study resource availability in mobile systems. The former observes no significant correlation within a single trip, but throughput traces show a higher degree of self-similarity during repeated trips. The latter paper, instead, classifies traffic according to spatial features and proposes a multi-class model to predict traffic, achieving promising results ($\varepsilon \sim 0.01$). Finally, although standard filtering techniques for static environments are less effective when applied to throughput of mobile nodes, they provide better accuracy when location is used as a context.

A.2. Bandwidth Availability Model

Based on the previous taxonomy, this section determines the main error sources and their impact on the statistical distribution of the predicted throughput. Figure A.1 shows examples of effects of errors on throughput prediction: the x-axis represents how far into the future the prediction is made, while the y-axis represents the predicted throughput and the corresponding estimation error. Note that purpose of the figure is to graphically exemplify the predictor categories; it is primarily intended to provide an intuition.

The figure examines the three categories of the taxonomy starting from (3)-*user* category on the left hand side. Here, the solid line represents the prediction itself, while the two dashed lines represent the confidence range of the prediction. Although the accuracy degrades with time, predictors belonging to this category are able to closely follow the throughput variations. As soon as the confidence range becomes as large as the signal's standard deviation, category (2)-*cell* predictors becomes as effective as category (3)-*user* predictors.

In the center, predictions obtained from the category (2)-*cell* are shown. Here, the predictions are averaged over longer time periods and their variability is represented by error bars. The solid line represents the actual prediction average along with its standard deviation, while the dashed line represents the same for the original signal. Predictors in this category infer user throughput from their position and statistics of the corresponding network cell.

Whenever it is not possible to predict the next user location, only predictors in category (1)-*net* can be used (right hand side of Figure A.1). They derive an estimate of user throughput from general network information using, for example, the generic distribution of user throughput in the overall network (shown in the figure as a dashed line).

To model the impact of errors on the predictors, we start from a simple formulation of the phenomenon itself. A very popular user throughput model can be found, for instance, in [124]. Here, the throughput T of a user with a distance of d kilometers to the transmitter and competing with N other users uniformly distributed within the coverage area of the transmitter, is represented as a function of the Signal to Interference plus Noise Ratio (SINR) γ, and N:

$$T = g_T(\Gamma, N) = T_0\eta/N, \tag{A.1}$$

where $\Gamma = 10\log_{10}\gamma$ is the SINR in dB, T_0 is a parameter specific to the actual cellular system and $\eta = g_\eta(\Gamma)$ is the spectral efficiency for that SINR. The SINR is a function of d and the fast fading gain r:

$$\gamma = g_\gamma(d, r) = \gamma_0 r/d^\alpha, \tag{A.2}$$

where γ_0 is a technology specific parameter and α is the pathloss exponent.

For what concerns errors themselves, different predictors are impacted by different error sources: for instance, those belonging to the third category try to model the short term behavior of the achievable throughput starting from past information. Thus, predicted throughput $\tilde{T} = T + e_T$, is the sum of the actual throughput and the prediction erro. Given that the error e_T has a probability density function (pdf) $f_{e_T}(e)$, the predicted throughput will have a pdf $f_{\tilde{T}} = f_{e_T}(e-T)$. Also, in the worst case the s-sample prediction can be modeled has the sum of s i.i.d random variables with distribution $f_{e_T}(e)$. Thus the s-sample predicted throughput distribution can be obtained as $f_{\tilde{T}(s)} = f_{e_T}((e-T)/s)/s$, which will have an expected value $\mu_{\tilde{T}(s)} = T(s) + s\mu_{e_T}$ and standard deviation $\sigma_{\tilde{T}(s)} = s\sigma_{e_T}$. Note that increasing s makes the prediction less and less accurate up to a point where the standard deviation of the prediction becomes comparable to the variability of the throughput σ_T.

Beyond this point using this type of predictors is useless and category 2 and category 1 predictors should be used. In this case, most of the predictors try to first estimate system parameters, such as the distance d and the number N of users and, from those, estimate the throughput distribution. Thus, in order to model the latter from the distributions of d and N, we will proceed as follows. First we analyse the distribution of the SINR given that N user are competing for the channel. It depends on the joint distribution $f_{r,d}(r, d|N)$, of the fading gain r and the distance d according to (A.2):

$$f_\gamma(\gamma|N) = \int_0^\infty f_{r,d}(g_\gamma^{-1}(\gamma, d), d|N)\left|\frac{\partial g_\gamma^{-1}(\gamma, d)}{\partial\gamma}\right|\mathrm{d}d, \tag{A.3}$$

where $g_\gamma^{-1}(\gamma, d)$ is the inverse function of (A.2) and we remove the variable d from the joint distribution $f_{\gamma,d}(\gamma, d|N)$ by integrating it on its whole support. Note that it is important to condition on N in order to account for opportunistic gain effects.

The last step requires to compute the throughput from the SINR using $g_\eta(\Gamma)$, which can be a piece-wise constant or other non-differentiable functions. In this case it is easier to use the

cumulative distribution functions (CDF), since we can avoid to use the derivative. In fact, the throughput CDF $F_T(x|N) = P(T \leq x) = P(g_T(\gamma) \leq x) = P(\gamma \leq g_T^{-1}(x)) = F_\gamma(\gamma^*|N)$, where $\gamma^* = g_T^{-1}(x)$. Thus,

$$F_T(x|N) = \int_0^{\gamma^*} f_\gamma(\gamma|N)\mathrm{d}\gamma. \tag{A.4}$$

The SINR and the throughput distributions can be obtained removing the dependency on N by multiplying by the probability mass function (pmf) of the number of user p_N, and summing over N. Thus,

$$F_\gamma(\gamma) = \sum_{i=1}^{M_N} p_i \int_0^{\gamma} f_\gamma(\gamma|i), \tag{A.5}$$

$$F_T(x) = \sum_{i=1}^{M_N} p_i F_T(x|i), \tag{A.6}$$

where M_N is chosen so that $p_{M_N} > 0$ and $p_{M_N+i} = 0, \forall i > 0$.

Note that, thanks to the independence of the fading and the distance distributions, their joint distribution can be written as the product of the two distributions:

$$f_{r,d}(r, d|N) = f_r(r|N) f_d(d). \tag{A.7}$$

It is easy to customize the model by modifying the distributions of three basic random variables, namely p_N, $f_d(d)$, $f_r(r|N)$. In particular, it is possible to include temporal and/or spatial dependencies by letting the distributions vary according to the location and the time.

A.3. Results

In this section we apply the model to the case of an LTE cellular system as defined in [125] adopting a Proportional Fair (PF) scheduler modeled according to the results presented in Sections II.D and III.B of [124].

In particular, we provide more specific definitions for some of the previous parameters: $T_0 = N_R B_R$, where N_R is the number of resource blocks and B_R is channel bandwidth; $\gamma_0 = 10^{(P_T - N_f + C)/10}$, where P_T is the eNodeB transmission power in dB N_f is the noise plus interference power in dB and $C = 128.1$ dB is a constant modeling other effects (such as antenna gains, frequency dependency, etc.); $g_\eta(\Gamma) = c_i$ if $G_i < \Gamma \leq G_{i+1}$ with $i \in \{0, \ldots, 15\}$ and $G_{16} = \infty$. c_i is the bit efficiency of the modulation of the i-th Modulation and Coding Scheme (MCS). The values for c_i and G_i are derived from [125] and are given in Table A.2.

In order to derive the exact expression for the SINR and the throughput distributions, we need to specify the distributions for the fading gain r, the distance d, between the user equipment and the eNodeB, and the number N, of user in the cell.

For what concerns the fading gain, we follow the results of [124], which models the oppor-

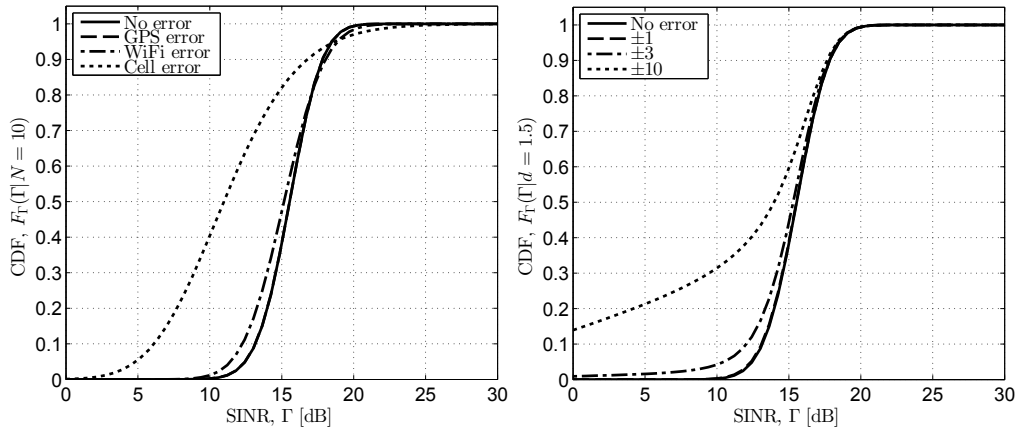


Figure A.2: Plots of the SINR CDF F_Γ, given a perfect knowledge of $N = 10$ (left) or a perfect knowledge of $d = 1.5$ Km (right). In the former case the standard deviation σ_d, of the distance is set as that of the most common localization systems, while in the latter $\sigma_N \in \{0, 1, 3, 10\}$.

Table A.2: MCS coefficients

CQI	Mod.	G_i	c_i
0	N/A	$-\infty$	0
1	QPSK	-6.00	0.15
2		-4.14	0.23
3		-2.29	0.38
4		-0.43	0.60
5		1.43	0.88
6		3.29	1.18
7	16QAM	5.14	1.48
8		7.00	1.91
9		8.86	2.41
10	64QAM	10.71	2.73
11		12.57	3.32
12		14.43	3.90
13		16.29	4.52
14		18.14	5.12
15		20.00	5.55

tunistic gain obtainable by the PF scheduler as follows:

$$f_r(r|N) = N(1 - e^{-r})^{N-1} e^{-r}. \tag{A.8}$$

This gain is associated to the higher probability for a user to be scheduled having a high SINR, when more users are competing for the channel.

The distance distribution $f_d(d)$, is obtained as the sum of two components: the actual distance distribution and the error committed in evaluating and/or predicting it. In the following, we anal-

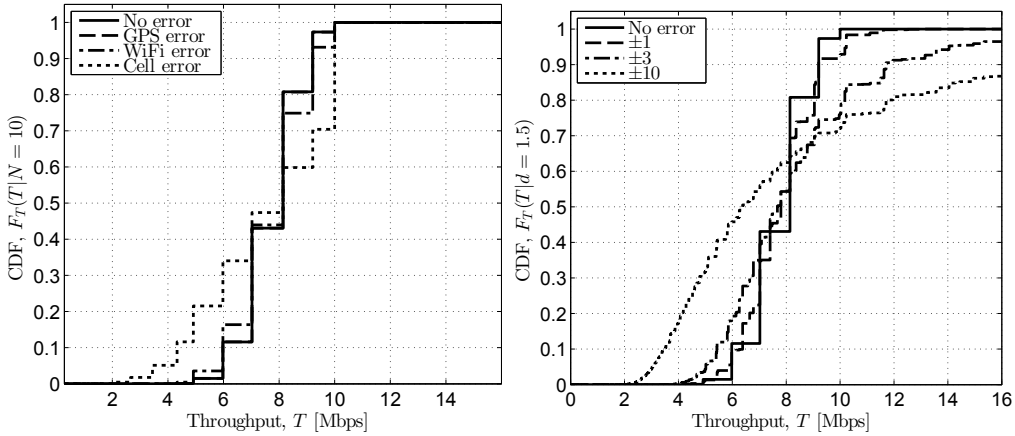


Figure A.3: Plots of the throughput CDF F_T, given a perfect knowledge of $N = 10$ (right left side) or a perfect knowledge of $d = 1.5$ Km (right hand side). In former case the standard deviation σ_d, of the distance is set as that of the most common localization systems, while in the latter, $\sigma_N \in \{0, 1, 3, 10\}$.

yse the case of a static user, whose distance is obtained with the three most common methods: GPS, WiFi and cell signal strength. In all the three cases we model the distance with a Gaussian distribution with an average $\mu_d = d^*$, equal to the correct user position d^*, and a standard deviation $\sigma_d = \{10, 100, 500\}$ meters, for GPS, WiFi and cell localization [119], respectively. Since the Gaussian distribution can lead to positive probability for negative values, we will normalize by $1 - \Phi(-\mu_d/\sigma_d)$, where $\Phi(x)$ is the CDF of a Gaussian distribution computed in x.

Similarly, the distribution of the number of users N, depends both on the actual value N^*, and the estimation error. As above, we take the Gaussian distribution as a reference:

$$p_i = \Phi((i - \mu_N)/\sigma_N)/(\sigma_N \sum_j p_j) \tag{A.9}$$

with $i \in \{1, \ldots, M_N\}$, where $\mu_N = N^*$ is the average value of the distribution and $\sigma_N \in \{0, 1, 3, 10\}$ are the standard deviation values we studied in the following examples of Figure A.2 and Figure A.3.

In these two examples, we focused on a single error at a time so as to separate the effects of an erroneous knowledge of N and d. The plot presents results for which we applied the aforementioned distributions to (A.3), obtaining:

$$f_\gamma(\gamma|N) = \int_0^\infty \frac{N d^\alpha}{\gamma_0 \sigma_d} \Big(1 - e^{-\frac{\gamma d^\alpha}{\gamma_0}}\Big)^{N-1} e^{-\frac{\gamma d^\alpha}{\gamma_0}} \phi\Big(\frac{d - \mu_d}{\sigma_d}\Big) \mathrm{d}d. \tag{A.10}$$

Now it is possible to compute γ^* as

$$\begin{aligned} \gamma^* &= 10^{g_\eta^{-1}(\frac{TN}{N_R B_R})/10} \\ \gamma_{i,N} &= 10^{G_{i+1}/10} \text{ with } \frac{c_i T_0}{N} \leq T < \frac{c_{i+1} T_0}{N}, \end{aligned} \tag{A.11}$$

which depends on both the bandwidth and the number of users. Now it is possible to compute (A.5) and (A.6), by using (A.10), (A.11) and (A.9).

In particular, Figure A.2 (left) shows $F_\Gamma(\Gamma|N = 10)$, using $f_d(d) = \mathcal{N}(\mu_d = d^* = 1.5, \sigma_d)$, and $\sigma_d \in \{0, 0.01, 0.1, 0.5\}$ to represent a static user, whose position is obtained with a localization error ranging from perfect knowledge to the worst approximation of a cell system localization. The figure shows that only with the precision of GPS is it possible to accurately estimate the statistical distribution of the SINR and that, if GPS information is lacking, the SINR prediction distribution becomes very wide even for static users.

Similarly, Figure A.2 (right) shows $F_\Gamma(\Gamma|d = 1.5)$ and the number of users distributed according to (A.9) using $\mu_N = 10$, $\sigma_N \in \{0, 1, 3, 10\}$ and $M_N = \mu_N + 5\sigma_N$. Again, for low σ_N, the distribution maintains the original shape, but as soon as $\sigma_N > 1$ the SINR distribution starts to get wider and is shifted towards the left. Note that, an error on N implies that $P(\tilde{\gamma} > \gamma) = 0$ $\forall\gamma$, which is a direct consequence of the modeling of the opportunistic gain of the PF scheduler.

The last two figures, Figure A.3 (left) and Figure A.3 (right), study F_T with errors on d and N, respectively. The error distributions are shaped as above, but this time the discontinuities of $g_\eta(\gamma)$ are evident. In particular, for a wider SINR distribution a larger number of MCS get positive probability of being used. Also, on the right hand side figure, the throughput CDF becomes smoother and smoother for increasing σ_N. This is due to the wider range of $\gamma_{i,N}$ introduced by (A.11).

Besides the trivial conclusion that the throughput distribution widens as the uncertainties grow, our model allows to compute where the correct value of the throughput is more likely to be found when a given prediction is computed. Also, the model allows to estimate the likelihood of the throughput to fall below a given threshold, thus enabling the study of resource allocation techniques when future information has limited reliability.

A.4. Summary

In this appendix we proposed a novel stochastic model for the user throughput prediction in mobile networks. The model takes into consideration the most relevant sources of prediction inexactness, such as random phenomena (e.g., fast fading) or imprecise information (e.g., user location). In fact the model derives the statistical distribution of the user throughput starting from those of the error sources. Also, the model allows for a closed form analysis of throughput prediction, which, in turn, enables to study the likelihood of forecasting-based optimization techniques to achieve their objective.

References

[1] N. Bui, F. Michelinakis, and J. Widmer, "Fine-grained lte radio link estimation for mobile phones," 2017.

[2] F. Kaup, F. Michelinakis, N. Bui, J. Widmer, K. Wac, and D. Hausheer, "Assessing the implications of cellular network performance on mobile content access," *IEEE Transactions on Network and Service Management*, vol. 13, no. 2, pp. 168–180, 2016.

[3] Cisco, "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016 - 2021 White Paper," *Cisco Public Information*.

[4] D. Wei, S. Rallapalli, R. Jama, L. Qiu, K. K. Ramakrishnan, L. Razoumov, Y. Zhang, and T. W. Cho, "ideal: Incentivized dynamic cellular offloading via auctions," in *Proceedings IEEE INFOCOM*, 2013.

[5] C. Joe-Wong, S. Sen, and S. Ha, "Complementary wireless network technologies: Adoption behavior and offloading benefits," *CoRR*, 2012.

[6] S. Wang, Y. Xin, S. Chen, W. Zhang, and C. Wang, "Enhancing spectral efficiency for lte-advanced and beyond cellular networks [guest editorial]," *IEEE Wireless Communications*, vol. 21, no. 2, pp. 8–9, April 2014.

[7] Z. Lu and G. de Veciana, "Optimizing stored video delivery for mobile networks: the value of knowing the future," in *IEEE INFOCOM 2013*, 2013, pp. 2706–2714.

[8] H. Abou-zeid, H. Hassanein, and S. Valentin, "Energy-efficient adaptive video transmission: Exploiting rate predictions in wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 5, pp. 2013–2026, June 2014.

[9] N. Bui and J. Widmer, "Mobile network resource optimization under imperfect prediction," in *Proc. IEEE WoWMoM*, June 2015.

[10] F. Michelinakis, "Practical challenges of network optimized stored video delivery," Master's thesis, Universidad Carlos III de Madrid, Spain, 2013.

[11] Y. Qiao, J. Skicewicz, and P. Dinda, "An empirical study of the multiscale predictability of network traffic," in *Proceedings IEEE HDCP*, 2004.

[12] N. Sadek and A. Khotanzad, "Multi-scale high-speed network traffic prediction using k-factor Gegenbauer ARMA model," in *Proceedings IEEE ICC*, 2004.

[13] N. Bui, I. Malanchini, and J. Widmer, "Anticipatory Admission Control and Resource Allocation for Media Streaming in Mobile Networks," in *Proc. ACM MSWIM*, Cancum, Mexico, November 2015.

[14] G. Kreitz and F. Niemelä, "Spotify – large scale, low latency, P2P music-on-demand streaming," in *Peer-to-Peer Computing*. IEEE, 2010, pp. 1–10.

[15] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, "A close examination of performance and power characteristics of 4G LTE networks," in *ACM MobiSys*, Low Wood Bay, Lake District, United Kingdom, June 2012, pp. 225–238.

[16] J. Huang, F. Qian, Y. Guo, Y. Zhou, Q. Xu, Z. M. Mao, S. Sen, and O. Spatscheck, "An in-depth study of LTE: Effect of network protocol and application behavior on performance," in *ACM SIGCOMM*, Hong Kong, China, August 2013, pp. 363–374.

[17] J. B. Landre, Z. E. Rawas, and R. Visoz, "Lte performance assessment prediction versus field measurements," in *IEEE PIMRC*, Sep. 2013, pp. 2866–2870.

[18] A. Nikravesh, D. R. Choffnes, E. Katz-Bassett, Z. M. Mao, and M. Welsh, "Mobile network performance from user devices: A longitudinal, multidimensional analysis," in *Springer PAM*, 2014, pp. 12–22.

[19] A. Nikravesh, H. Yao, S. Xu, D. Choffnes, and Z. M. Mao, "Mobilyzer: An open platform for controllable mobile network measurements," in *ACM MobiSys*, 2015, pp. 389–404.

[20] N. Bui and J. Widmer, "OWL: a Reliable Online Watcher for LTE Control Channel Measurements," in *ACM All Things Cellular*, Oct. 2016.

[21] U. Goel, M. Steiner, M. P. Wittie, M. Flack, and S. Ludin, "Http/2 performance in cellular networks," in *ACM MobiCom*, 2016.

[22] E. Nygren, R. K. Sitaraman, and J. Sun, "The akamai network: a platform for high-performance internet applications," *ACM SIGOPS Operating Systems Review*, vol. 44, no. 3, pp. 2–19, 2010.

[23] N. Bui, F. Michelinakis, and J. Widmer, "A model for throughput prediction for mobile users," in *European Wireless*, Barcelona, Spain, May 2014.

[24] F. Michelinakis, N. Bui, G. Fioravantti, J. Widmer, F. Kaup, and D. Hausheer, "Lightweight mobile bandwidth availability measurement," in *IFIP Networking Conference*, May 2015.

[25] F. Michelinakis, G. Kreitz, R. Petrocco, B. Zhang, and J. Widmer, "Passive mobile bandwidth classification using short lived tcp connections," in *WMNC*, 2015.

[26] F. Kaup, F. Michelinakis, N. Bui, J. Widmer, K. Wac, and D. Hausheer, "Behind the NAT – A measurement based evaluation of cellular service quality," in *CNSM*, 2015.

[27] F. Michelinakis, H. Doroud, A. Razaghpanah, A. Lutu, N. Vallina-Rodriguez, P. Gill, and J. Widmer, "The Cloud that Runs the Mobile Internet: A Measurement Study of Mobile Cloud Services," in *Proc. IEEE INFOCOM*, Honolulu, HI, USA, April 2018.

[28] F. Michelinakis, N. Bui, G. Fioravantti, F. Kaup, D. Hausheer, and J. Widmer, "Lightweight capacity measurements for mobile networks," *Elsevier Computer Communications*, vol. 84, pp. 73–83, Jun. 2016.

[29] C. Koch, N. Bui, J. Rückert, G. Fioravantti, F. Michelinakis, S. Wilk, J. Widmer, and D. Hausheer, "Media download optimization through prefetching and resource allocation in mobile networks," in *Proceedings of the 6th ACM Multimedia Systems Conference*. ACM, 2015, pp. 85–88.

[30] C. Koch, J. Ruckert, N. Bui, F. Michelinakis, G. Fioravantti, D. Hausheer, and J. Widmer, "Mobile social prefetcher using social and network information," 2014.

[31] F. Michelinakis, "Mobile capacity measurements and estimation," 2014.

[32] M. Marciel, F. Michelinakis, R. Fanou, and P. J. Muñoz-Merino, "Enhancements to google course builder: Assessments visualisation, youtube events collector and dummy data generator," 2013.

[33] 3GPP release 15: The first set of 5G standards. [Online]. Available: http://www.3gpp.org/release-15

[34] Alcatel-Lucent, "STRATEGIC WHITE PAPER: The LTE Network Architecture: A comprehensive tutorial," Tech. Rep., 2013, http://www.cse.unt.edu/~rdantu/FALL_2013_WIRELESS_NETWORKS/LTE_Alcatel_White_Paper.pdf.

[35] C. Johnson, *Long Term Evolution IN BULLETS*, 2nd ed. CreateSpace Independent Publishing Platform, 2012.

[36] S. Chen, S. Sun, Y. Wang, G. Xiao, and R. Tamrakar, "A comprehensive survey of TDD-based mobile communication systems from TD-SCDMA 3G to TD-LTE(A) 4G and 5G directions," *Communications, China*, vol. 12, no. 2, pp. 40–60, Feb 2015.

[37] R. Kwan, C. Leung, and J. Zhang, "Proportional fair multiuser scheduling in LTE," *Signal Processing Letters, IEEE*, vol. 16, no. 6, pp. 461–464, 2009.

[38] F. Zarinni, A. Chakraborty, V. Sekar, S. R. Das, and P. Gill, "A first look at performance in mobile virtual network operators," in *Proceedings of the 2014 Conference on Internet Measurement Conference*. ACM, 2014, pp. 165–172.

[39] (last accessed June 2018) Common Android Kernel Tree. [Online]. Available: https://android.googlesource.com/kernel/common/

[40] (last accessed June 2018) Definition of the skbuf structure in Android source code. [Online]. Available: https://android.googlesource.com/kernel/common/+/a7827a2a60218b25f222b54f77ed38f57aebe08b/include/linux/skbuff.h

[41] (last accessed June 2018) Explanation of some fields of skbuf structure. [Online]. Available: http://vger.kernel.org/~davem/skb.html

[42] "Scaling in the linux networking stack," https://android.googlesource.com/kernel/msm/+/android-wear-5.0.2_r0.1/Documentation/networking/scaling.txt, last accessed June 2018.

[43] E. Dumazet, "Busy polling: Past, present, future."

[44] (last accessed June 2018) BUSY POLLING Netdev 2.1. [Online]. Available: https://netdevconf.org/2.1/slides/apr6/dumazet-BUSY-POLLING-Netdev-2.1.pdf

[45] Ookla, "Ookla speedtest mobile apps," http://www.speedtest.net/mobile/, last accessed June 2014.

[46] Y. Xu, Z. Wang, W. K. Leong, and B. Leong, "An end-to-end measurement study of modern cellular data networks," in *Passive and Active Measurement*. Springer, 2014, pp. 34–45.

[47] K. Lai and M. Baker, "Measuring link bandwidths using a deterministic model of packet delay," in *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, ser. ACM SIGCOMM '00, New York, NY, USA, pp. 283–294. [Online]. Available: http://doi.acm.org/10.1145/347059.347557

[48] C. Dovrolis, P. Ramanathan, and D. Moore, "Packet-dispersion techniques and a capacity-estimation methodology," *IEEE/ACM Transactions on Networking*, vol. 12, no. 6, pp. 963–977, December 2004.

[49] R. Kapoor, L.-J. Chen, L. Lao, M. Gerla, and M. Sanadidi, "CapProbe: a simple and accurate capacity estimation technique," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 4, pp. 67–78, October 2004.

[50] A. Gerber, J. Pang, O. Spatscheck, and S. Venkataraman, "Speed testing without speed tests: estimating achievable download speed from passive measurements," in *ACM IMC*, Melbourne, Australia, November 2010, pp. 424–430.

[51] Q. Xu, S. Mehrotra, Z. Mao, and J. Li, "PROTEUS: network performance forecast for real-time, interactive mobile applications," in *ACM MobiSys*, Taipei, Taiwan, June 2013, pp. 347–360.

[52] F. Ricciato, F. Vacirca, and M. Karner, "Bottleneck Detection in UMTS via TCP Passive Monitoring: A Real Case," in *ACM CoNEXT*, Toulouse, France, October 2005, pp. 211–219.

[53] P. Svoboda and F. Ricciato, "Analysis and detection of bottlenecks via TCP footprints in live 3G networks," in *IFIP WiOPT*, Hammammet, Tunisia, April 2008, pp. 37–42.

[54] C. Ide, B. Dusza, and C. Wietfeld, "Performance of channel-aware M2M communications based on LTE network measurements." in *IEEE PIMRC*, 2013, pp. 1614–1618.

[55] X. Xie, X. Zhang, S. Kumar, and L. E. Li, "pistream: Physical layer informed adaptive video streaming over lte," in *ACM MobiCom*, Sep. 2015, pp. 413–425.

[56] S. Kumar, E. Hamed, D. Katabi, and L. Erran Li, "LTE radio analytics made easy and accessible," in *ACM SIGCOMM*, vol. 44, no. 4, 2014, pp. 211–222.

[57] Y. Li, C. Peng, Z. Yuan, J. Li, H. Deng, and T. Wang, "Mobileinsight: Extracting and analyzing cellular network information on smartphones," in *ACM Mobicom*, Oct. 2016.

[58] W. Li, R. K. P. Mok, D. Wu, and R. K. C. Chang, "On the accuracy of smartphone-based mobile network measurement," in *IEEE INFOCOM*, Hong Kong, April 2015, pp. 370–378.

[59] A. Elnashar and M. A. El-Saidny, "Looking at lte in practice: A performance analysis of the lte system based on field test results," *IEEE Vehicular Technology Magazine*, vol. 8, no. 3, pp. 81–92, Sep. 2013.

[60] N. Becker, A. Rizk, and M. Fidler, "A measurement study on the application-level performance of LTE," in *IFIP Networking Conference*, 2014, pp. 1–9.

[61] M. Jovanovic, M. K. Karray, and B. Blaszczyszyn, "QoS and network performance estimation in heterogeneous cellular networks validated by real-field measurements," in *ACM PE-WASUN*, 2014, pp. 25–32.

[62] N. Vallina-Rodriguez, S. Sundaresan, C. Kreibich, N. Weaver, and V. Paxson, "Beyond the radio: Illuminating the higher layers of mobile networks," in *ACM MobiSys*, 2015.

[63] "During Netflix money fight, Cogent's other big customers suffered too." [Online]. Available: https://arstechnica.com/information-technology/2014/11/during-netflix-money-fight-cogents-other-big-customers-suffered-too/

[64] A. Molavi Kakhki, A. Razaghpanah, A. Li, H. Koo, R. Golani, D. Choffnes, P. Gill, and A. Mislove, "Identifying traffic differentiation in mobile networks," in *Proceedings of the 2015 Internet Measurement Conference*, ser. IMC '15. New York, NY, USA: ACM, 2015, pp. 239–251. [Online]. Available: http://doi.acm.org/10.1145/2815675.2815691

[65] J. P. Rula and F. E. Bustamante, "Behind the curtain: Cellular dns and content replica selection," in *Proc. IMC*. ACM, 2014, pp. 59–72.

[66] J. S. Otto, M. A. Sánchez, J. P. Rula, and F. E. Bustamante, "Content delivery and the natural evolution of dns: remote dns trends, performance issues and alternative solutions," in *Proc. IMC*. ACM, 2012, pp. 523–536.

[67] B. Frank, I. Poese, Y. Lin, G. Smaragdakis, A. Feldmann, B. Maggs, J. Rake, S. Uhlig, and R. Weber, "Pushing CDN-ISP Collaboration to the Limit," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 3, pp. 34–44, Jul. 2013. [Online]. Available: http://doi.acm.org/10.1145/2500098.2500103

[68] I. N. Bozkurt, A. Aguirre, B. Chandrasekaran, P. B. Godfrey, G. Laughlin, B. Maggs, and A. Singla, "Why is the internet so slow?!" in *International Conference on Passive and Active Network Measurement*. Springer, 2017, pp. 173–187.

[69] C. Contavalli, W. van der Gaast, D. Lawrence, and W. Kumari, "Client Subnet in DNS Queries," RFC 7871 (Informational), Internet Engineering Task Force, May 2016. [Online]. Available: http://www.ietf.org/rfc/rfc7871.txt

[70] S. Narayana, J. W. Jiang, J. Rexford, and M. Chiang, "Distributed wide-area traffic management for cloud services," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, no. 1. ACM, 2012, pp. 409–410.

[71] R. Krishnan, H. V. Madhyastha, S. Srinivasan, S. Jain, A. Krishnamurthy, T. Anderson, and J. Gao, "Moving beyond end-to-end path information to optimize cdn performance," in *ACM IMC*, 2009.

[72] U. Goel, M. Steiner, M. P. Wittie, M. Flack, and S. Ludin, "Measuring what is not ours: A tale of 3rd party performance," in *International Conference on Passive and Active Network Measurement*. Springer, 2017, pp. 142–155.

[73] MartinGarcia, Luis, "Tcpdump," http://www.tcpdump.org/, last accessed November 2016.

[74] S. Bauer, D. D. Clark, and W. Lehr, "Understanding broadband speed measurements." TPRC, 2010.

[75] Netflix, https://help.netflix.com/en/node/306, last accessed June 2015.

[76] "The network simulator - ns-3," http://www.nsnam.org/, last accessed September 2015.

[77] "LENA - ns-3 LTE module," http://lena.cttc.es/manual/, last accessed September 2015.

[78] A. Razaghpanah, N. Vallina-Rodriguez, S. Sundaresan, C. Kreibich, P. Gill, M. Allman, and V. Paxson, "Haystack: A multi-purpose mobile vantage point in user space," *arXiv preprint arXiv:1510.01419v3*, 2016.

[79] Ö. Alay, A. Lutu, M. Peón-Quirós, V. Mancuso, T. Hirsch, T. Dely, J. Werme, K. Evensen, A. Hansen, S. Alfredsson, J. Karlsson, A. Brunstrom, A. Khatouni, M. Mellia, and M. Marsan, "Experience: An Open Platform for Experimentation with Commercial Mobile Broadband Networks," in *Proc. of ACM Mobicom*, 2017.

[80] "Akamai Facts & Figures," 2017, https://www.akamai.com/uk/en/about/facts-figures.jsp.

[81] "AWS Global Infrastructure," https://aws.amazon.com/about-aws/global-infrastructure/.

[82] D. Wu, R. K. Chang, W. Li, E. K. Cheng, and D. Gao, "Mopeye: Opportunistic monitoring of per-app mobile network performance," *arXiv preprint arXiv:1703.07551*, 2017.

[83] I. Castro, J. C. Cardona, S. Gorinsky, and P. Francois, "Remote peering: More peering without internet flattening," in *Proc. CoNEXT*. ACM, 2014, pp. 185–198.

[84] A. Dhamdhere and C. Dovrolis, "The internet is flat: modeling the transition from a transit hierarchy to a peering mesh," in *Proc. CoNEXT*. ACM, 2010, p. 21.

[85] "Netflix Open Connect." [Online]. Available: https://openconnect.netflix.com

[86] "Akamai Network Partnerships." [Online]. Available: https://www.akamai.com/uk/en/products/network-operator/akamai-network-partnerships.jsp

[87] V. K. Adhikari, Y. Guo, F. Hao, V. Hilt, Z.-L. Zhang, M. Varvello, and M. Steiner, "Measurement study of netflix, hulu, and a tale of three cdns," *IEEE/ACM ToN*, vol. 23, no. 6, pp. 1984–1997, 2015.

[88] N. Vallina-Rodriguez, S. Sundaresan, A. Razaghpanah, R. Nithyanand, M. Allman, C. Kreibich, and P. Gill, "Tracking the Trackers: Towards Understanding the Mobile Advertising and Tracking Ecosystem," in *Workshop on Data and Algorithmic Transparency*, Nov. 2016.

[89] A. Razaghpanah, R. Nithyanand, N. Vallina-Rodriguez, S. Sundaresan, M. Allman, C. Kreibich, and P. Gill, "Apps, Trackers, Privacy, and Regulators: A Global Study of the Mobile Tracking Ecosystem," in *Proc. NDSS*, 2018.

[90] ICSI, "Lumen Privacy Monitor," 2016, https://play.google.com/store/apps/details?id=edu.berkeley.icsi.haystack.

[91] "CDNFinder by CDNPlanet," https://www.cdnplanet.com/tools/cdnfinder/.

[92] Intel Security/McAfee, "Customer URL Ticketing System," http://www.trustedsource.org/.

[93] "OpenDNS Domain Tagging," https://domain.opendns.com.

[94] "Cloudmap Project," 2018, http://wireless.networks.imdea.org/cloudmap-project.

[95] "edns-client-subnet participants," http://www.afasterinternet.com/participants.htm.

[96] E. Halepovic et. al, "Can You GET Me Now?: Estimating the Time-to-first-byte of HTTP Transactions with Passive Measurements," in *Proc. ACM IMC*, 2012.

[97] N. Vallina-Rodriguez, J. Shah, A. Finamore, Y. Grunenberger, K. Papagiannaki, H. Haddadi, and J. Crowcroft, "Breaking for Commercials: Characterizing Mobile Advertising," in *Proceedings of the Internet Measurement Conference*, 2012.

[98] ICSI, "Netalyzr," http://netalyzr.icsi.berkeley.edu/.

[99] C. Kreibich, N. Weaver, B. Nechaev, and V. Paxson, "Netalyzr: illuminating the edge network," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. ACM, 2010, pp. 246–259.

[100] A. Razaghpanah, A. A. Niaki, N. Vallina-Rodriguez, S. Sundaresan, J. Amann, and P. Gill, "Studying TLS Usage in Android Apps," in *Proc. ACM CoNEXT*, 2017.

[101] M. Almeida, A. Finamore, D. Perino, N. Vallina-Rodriguez, and M. Varvello, "Dissecting DNS Stakeholders in Mobile Networks," in *Proc. ACM CoNEXT*, 2017.

[102] Y. Shavitt and N. Zilberman, "A geolocation databases study," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 10, pp. 2044–2056, 2011.

[103] CAIDA/UCSD, "Mapping Autonomous Systems to Organizations: CAIDA's Inference Methodology." [Online]. Available: https://www.caida.org/research/topology/as2org/

[104] J. P. Rula, F. E. Bustamante, and M. Steiner, "Cell spotting: studying the role of cellular networks in the internet," in *Proc. ACM IMC*, 2017.

[105] CAIDA/UCSD, "AS Relationships Dataset." [Online]. Available: http://as-rank.caida.org/

[106] T. Sonera, "Telia and Akamai announce strategic relationship to deliver enhanced web services throughout Europe." [Online]. Available: https://www.teliacompany.com/en/news/press-releases/2000/1/telia-and-akamai-announce-strategic-relationship-to-deliver-enhanced-web-services-throughout-europe/

[107] Akamai Press Release, "Akamai And Telecom Italia Enter Into Partnership To Offer Content Delivery And Web Optimization Solutions." [Online]. Available: https://www.akamai.com/us/en/about/news/press/2015-press/akamai-and-telecom-italia-enter-into-partnership-to-offer-content-delivery-and-web-optimization-solutions.jsp

[108] (last accessed October 2015) Alexa: Most popular websites in Germany. [Online]. Available: http://www.alexa.com/topsites/countries/DE

[109] A. Burulitisz, S. Imre, and S. Szabó, "On the accuracy of mobility modelling in wireless networks," in *Communications, 2004 IEEE International Conference on*, vol. 4. IEEE, 2004, pp. 2302–2306.

[110] K. Papagiannaki, N. Taft, Z.-L. Zhang, and C. Diot, "Long-term forecasting of internet backbone traffic: Observations and initial models," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, vol. 2. IEEE, 2003, pp. 1178–1188.

[111] B. Zhou, D. He, Z. Sun, and W. H. Ng, "Network traffic modeling and prediction with arima/garch," in *HET-NETs 06 Conference*. Citeseer, 2005, pp. 1–10.

[112] L. Song, D. Kotz, R. Jain, and X. He, "Evaluating location predictors with extensive wi-fi mobility data," in *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, vol. 2. IEEE, 2004, pp. 1414–1424.

[113] W. Creixell and K. Sezaki, "Routing protocol for ad hoc mobile networks using mobility prediction," *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 2, no. 3, pp. 149–156, 2007.

[114] J. Froehlich and J. Krumm, "Route prediction from trip observations," *SAE SP*, vol. 2193, p. 53, 2008.

[115] W. Dong, N. Duffield, Z. Ge, S. Lee, and J. Pang, "Modeling cellular user mobility using a leap graph," in *Passive and Active Measurement*. Springer, 2013, pp. 53–62.

[116] J. Yao, S. S. Kanhere, and M. Hassan, "An empirical study of bandwidth predictability in mobile computing," in *ACM WiNTECH Workshop*, San Francisco, CA, USA, September 2008, pp. 11–18.

[117] A. J. Nicholson and B. D. Noble, "Breadcrumbs: forecasting mobile connectivity," in *Proceedings of the 14th ACM international conference on Mobile computing and networking*. ACM, 2008, pp. 46–57.

[118] M. Z. Shafiq, L. Ji, A. X. Liu, and J. Wang, "Characterizing and modeling internet traffic dynamics of cellular devices," in *Proceedings ACM SIGMETRICS*, 2011.

[119] P. A. Zandbergen, "Accuracy of iphone locations: A comparison of assisted gps, wifi and cellular positioning," *Transactions in GIS*, vol. 13, no. s1, pp. 5–25, 2009.

[120] M. C. Gonzalez, C. A. Hidalgo, and A.-L. Barabasi, "Understanding individual human mobility patterns," *Nature*, vol. 453, no. 7196, pp. 779–782, 2008.

[121] C. Song, Z. Qu, N. Blumm, and A.-L. Barabási, "Limits of predictability in human mobility," *Science*, vol. 327, no. 5968, pp. 1018–1021, 2010.

[122] V. A. Siris and D. Kalyvas, "Enhancing mobile data offloading with mobility prediction and prefetching," in *Proceedings of the seventh ACM international workshop on Mobility in the evolving internet architecture*, ser. MobiArch '12. New York, NY, USA: ACM, 2012, pp. 17–22. [Online]. Available: http://doi.acm.org/10.1145/2348676.2348682

[123] U. Paul, A. P. Subramanian, M. M. Buddhikot, and S. R. Das, "Understanding traffic dynamics in cellular data networks," in *INFOCOM, 2011 Proceedings IEEE*. IEEE, 2011, pp. 882–890.

[124] O. Osterbo, "Scheduling and capacity estimation in LTE," in *Teletraffic Congress (ITC), 2011 23rd International*, 2011, pp. 63–70.

[125] S. Sesia, I. Toufik, and M. Baker, *LTE: the UMTS long term evolution*. Wiley Online Library, 2009.