universität wien

DISSERTATION

Titel der Dissertation

„Measurement and Models of Video Streaming and Mobile Core Networks"

Verfasser

Dipl.-Inform. Florian Metzger

angestrebter akademischer Grad

Doktor der technischen Wissenschaften (Dr. tech.)

Wien, 2014

Studienkennzahl lt. Studienblatt:	A 786 880
Dissertationsgebiet lt. Studienblatt:	Informatik
Betreuer:	

Version Information:
commit (None)
from (None)

CONTENTS

FIGURES

TABLES

ACRONYMS

3G Third Generation. iii, vi, 37, 57, 87
3GPP Third Generation Partnership Project. i, vi, 16, 37, 39, 40, 42, 45, 46, 52, 57

AAA Authentication, Authorization and Accounting. vi, 37, 38
API Application Programming Interface. vi, 7
APN Access Point Name. vi, 43
ASN.1 Abstract Syntax Notation One. vi
ATM Asynchronous Transfer Mode. vi, 42
AU Access Unit. vi, 13
AuC Authentication Centre. vi

BMC Broadcast/Multicast Control. vi, 42
BSC Base Station Controller. vi
BSS Base Station Subsystem. vi, 40
BTS Base Transceiver Station. vi

CAMEL Customised Applications for Mobile Networks Enhanced Logic. vi, 43, 48
CDF Cumulative Distribution Function. vi, 85
CDMA Code Division Multiple Access. vi, 3
CDN Content Distribution Network. vi, 16, 18
CI Continuity Index. vi, 21
CN Core Network. vi
CS Circuit Switching. vi, 40
CSS Cascading Style Sheets. vi, 1

DASH Dynamic Adaptive Streaming over HTTP. vi, 12, 16, 29
DCCP Datagram Congestion Control Protocol. vi, 12
DDoS Distributed Denial of Service. vi, 38
DES Discrete Event Simulation. vi, 5, 35, 84
DF Delay Factor. vi, 20
DNS Domain Name System. vi, 16
DOCSIS Data Over Cable Service Interface Specification. vi, 37
DoS Denial of Service. vi, 55
DPI Deep Packet Inspection. vi, 43
DRM Digital Rights Management. vi

E-UTRAN Evolved UTRAN. vi, 40
ECM EPS Connection Management. vi
EDGE Enhanced Data Rates for GSM Evolution. vi, vii, 40, 62
EIR Equipment Identity Register. vi
EMM Evolved Mobility Management. vi
eNB Evolved Node B. vi
EPC Evolved Packet Core. vi, 42–45
EPS Evolved Packet System. vi, 43

ES Elementary Stream. vi, 13
ETSI European Telecommunications Standards Institute. vi, 39

FOSS Free and open-source software. vi
FTW Telecommunications Research Center Vienna. vi

GERAN Global System for Mobile Communications (GSM)/Enhanced Data Rates for GSM Evolution (EDGE) Radio Access Network. vi, 40
GGSN Gateway GPRS Support Node. iii, iv, vi, 39, 40, 42, 43, 46–50, 52, 55, 57, 58, 61, 62, 67, 72, 73, 75, 76, 79–86, 90, 91
GMSC Gateway MSC. vi
GPRS General Packet Radio System. vi, 3, 39, 40, 42, 43, 45, 46, 50, 52, 62, 87
GPS Global Positioning System. vi, 57, 58
GSM Global System for Mobile Communications. vi, vii, 3, 37, 39, 40, 45, 46, 59
GSMA GSM Association. vi, 59
GTP GPRS Tunneling Protocol. vi, 39, 42, 43, 45–47, 50–52, 55, 57–60, 67, 75, 76, 80, 87–89
GTPv2 GPRS Tunneling Protocol (GTP) version 2. vi

HLR Home Location Register. vi, 42, 61
HLS HTTP Live Streaming. vi, 16
HSDPA High-Speed Downlink Packet Access. vi, 40
HSPA High Speed Packet Access. vi, 4, 57, 62
HSPA+ High Speed Packet Access Plus. vi, 4, 40, 62
HSS Home Subscriber Server. vi, 42, 43
HSUPA High Speed Uplink Packet Access. vi, 40
HTML HyperText Markup Language. vi, 1
HTTP HyperText Transfer Protocol. vi, x, 9, 12–18, 24, 29, 30, 34

IAB Internet Architecture Board. vi
ICE Interactive Connectivity Establishment. vi, 13
ICMP Internet Control Message Protocol. vi
IE Information Element. vi, 47, 48, 50, 51, 89
IEC International Electrotechnical Commission. vi, 16
IEEE Institute of Electrical and Electronics Engineers. vi
IETF Internet Engineering Task Force. vi, 5, 40
IGMP Internet Group Management Protocol. vi, 13
IMEI International Mobile Equipment Identity. vi, 57–59
IMS IP Multimedia Subsystem. vi, 16
IMSI International Mobile Subscriber Identity. vi, 47, 50, 57, 58
IP Internet Protocol. vi, vii, 19, 37, 42, 43
IPTV Internet Protocol television. vi, 20
IPv4 IP version 4. vi
IPv6 IP version 6. vi
ISDN Integrated Services Digital Network. vi, 3
ISO International Organization for Standardization. vi, 5, 16
ISOC Internet Society. vi
ISP Internet Service Provider. vi, 18

ITU International Telecommunication Union. vi, 5, 20

KISS "Keep it simple, stupid". vi, 37

LTE Long Term Evolution. vi, 4, 37, 39, 40, 42, 46, 90

MAC Media Access Control. vi, 42
MAP Mobile Application Part. vi, 42, 43
MBMS Multimedia Broadcast Multicast Services. vi, 16
MDI Media Delivery Index. vi, 20
METAWIN Measurement and Traffic Analysis in Wireless Networks. iii, vi, 56–58
MGW Media Gateway. vi
MME Mobility Management Entity. vi, 43
MMS Microsoft Media Server. vi
MOS Mean Opinion Score. vi, 18, 19
MS Mobile Station. vi, 40, 45–47, 50, 57, 58
MS-ID Mobile Station Identifier. vi, 57–59, 94
MSC Mobile Switching Center. vi, vii
MSE Mean Squared Error. vi, 20
MSISDN Mobile Subscriber Integrated Services Digital Network-Number. vi
MTU Maximum Transmission Unit. vi

NAT Network Address Translation. vi, ix, 13
NBAP Node B Application Part. vi
NFV Network Function Virtualization. vi, 79, 80, 82, 83
NSAPI Network Service Access Point Identifier. vi, 51, 93

OS Operating System. vi, 62, 68

P2P Peer-to-Peer. vi, 16, 21
PCC Policy and Charging Control. vi
PCEF Policy and Charging Enforcement Function. vi, 43, 44
PCRF Policy and Charging Rules Function. vi, 43
PDCP Packet Data Convergence Protocol. vi, 42
PDN Public Data Network. vi, 43
PDP Packet Data Protocol. ii, vi, 39, 43, 47, 50–53, 55–57, 66, 76, 88
PEVQ Perceptual Evaluation of Video Quality. vi
PGW Packet Gateway. vi, 43
PI Pause Intensity. vi, 21
PLMN Public Land Mobile Network. vi, 52
PMIP Proxy Mobile IPv6. vi, 43
POTS Plain Old Telephone Service. vi, 3
PS Packet Switching. vi, 40, 42
PSNR Peak Signal-to-Noise Ratio. vi, 20
PSTN Public Switched Telephone Network. vi

QoE Quality of Experience. vi, 18, 20, 22
QoS Quality of Service. vi, 9, 15, 18, 20, 21, 23, 28, 29, 31, 32, 35, 47, 49, 50

RAB Radio Access Bearer. vi, 42, 46, 52, 61

RAN Radio Access Network. vi
RANAP Radio Access Network Application Part. vi, 42, 48
RAT Radio Access Technology. vi, 52, 53, 57, 58
RFC Request for Comments. vi
RLC Radio Link Control. vi, 42
RNC Radio Network Controller. vi, 42, 45, 46, 50, 55, 57, 61
RRC Radio Resource Control. iii, vi, 42, 46, 51, 52, 54, 55, 62, 65
RTCP RTP Control Protocol. vi, 12, 13
RTMP Real Time Messaging Protocol. vi, 16
RTP Real-time Transport Protocol. vi, 3, 12–14
RTSP Real Time Streaming Protocol. vi, 12, 13
RTT Round-Trip Time. vi, 32

S1AP S1 Application Protocol. vi, 43
SAE System Architecture Evolution. vi
SAP Session Announcement Protocol. vi, 13
SCCP Signalling Connection Control Part. vi
SCTP Stream Control Transmission Protocol. vi
SDN Software Defined Networking. vi
SDP Session Description Protocol. vi, 13
SDR Software Defined Radio. vi
SGSN Serving GPRS Support Node. iii, vi, 39, 40, 42, 43, 46–50, 52, 55, 57, 61
SGW Serving Gateway. vi, 43
SIP Session Initiation Protocol. vi, 13
SS7 Signalling System No. 7. vi
SSD Solid State Disk. vi, 86, 87
SSIM Structural SIMilarity. vi
STUN Session Traversal Utilities for NAT. vi, 13

TAC Type Allocation Code. v, vi, 57–61, 66, 94
TCAP Transaction Capabilities Application Part. vi
TCP Transmission Control Protocol. vi, 7, 9, 11, 13–16, 19–21, 30, 32, 33
TFT Traffic Flow Template. vi, 44
TS Technical Specification. vi, 40, 45, 46, 52
TTI Transmission Time Interval. vi, 75

UDP User Datagram Protocol. vi, 11–14
UE User Equipment. vi, 40, 42
UMTS Universal Mobile Telecommunications System. iii, vi, ix, 4, 15, 37–40, 42–46, 50, 57, 62, 73, 87
URL Uniform Resource Locator. vi, 26
USIM Universal Subscriber Identity Module. vi
UTRAN UMTS Terrestrial Radio Access Network. vi, 40, 46

VDSL Very-high-bit-rate Digital Subscriber Line. vi, 37
VQEG Video Quality Experts Group. vi, 20

W3C World Wide Web Consortium. vi, 26
WebRTC Web Real-Time Communication. vi

WMSP Windows Media HTTP Streaming Protocol. vi

XMPP Extensible Messaging and Presence Protocol. vi, 13

"If the Internet teaches us anything, it is that great value comes from leaving core resources in a commons, where they're free for people to build upon as they see fit."

– Lawrence Lessig

ABSTRACT

Short summary of the contents in English...

ZUSAMMENFASSUNG

Kurze Zusammenfassung des Inhaltes in deutscher Sprache...

ACKNOWLEDGMENTS

1 INTRODUCTION

Packet Switching, the process of chunking information into smaller bits, labeling it and transmitting it independently, was first described by Paul Baran in the 1960s[Bar64]. This changed communication a lot and provided one of the foundations for the emergence of the Internet.

The original usage intent was mostly limited to remotely logging into and communicating with mainframe computers and transferring files supported by the emergence of new multiuser and multitasking operating systems, especially including the release of UNIX in 1969.[TODO: needs references]

From these early times countless other services and applications emerged. Today, the Internet is by used hundreds of millions of people, touching almost every aspect of daily life. The creation of the World Wide Web – begun by Tim Berners-Lee in 1989[TODO: reference] – brought an easily accessible "user interface" to the Internet and made adoption for the masses much easier. Today, almost any form of application is available through the Web as a combination of HyperText Markup Language (HTML), Cascading Style Sheets (CSS), and JavaScript and can be accessed by just opening a Web browser.

But as is the case with many technological fields, the Internet's development was also closely entangled with other advancements. Take the huge Internet music wave beginning in 1999 as an example. It may have been started by the creation of Napster, the first peer-to-peer file sharing service. However, without the large improvements in audio compression – namely MP3 – a few years before, including computers that could handle decompression as well as compression and increasing Internet access bandwidth (and fixed service fees) , this would not have been possible. Similarly, YouTube could not have existed without a wide spread of digital camcorders and many other devices with recording capability, good video compression codes, and an even further increase in access bandwidths. For every leap in bandwidth new services sprung to life fueling the users' demand for further capacity increases.

Several factors contributed to the quick acceptance of the Internet. First and foremost no assumptions are made on the transported content, or content agnosticism. The original idea is to treat every packet, every participating node the same and do not make any assumptions on specific applications. This provides a level playing field for every contender wanting to offer services. This is upheld by the second driving concept, the "End-to-End Principle", [SRC84] stating:

> "The function in question can completely and correctly be implemented only with the knowledge and help of the application standing at the endpoints of the communications system. Therefore, providing that questioned function as a feature of the communication system itself is not possible. [...]"

End-to-End means that any functionality, meaning services or applications, should only be implemented at the two endpoints of a transmission not somewhere in between. Tuning the network to specific applications and implementing functions inside the network, will always only be valid for existing and narrow use cases and cannot support future developments.

During the past decades this has often been discussed [BCZ97; BC01; Ise97; LL00] but is still being upheld for the most part for good reason. [TODO: probably should clarify the good reason]. This is in contrast to the way telephone networks and their operating companies historically worked. These networks are based on circuit switching and access as well as content (namely voice calls) and procedures are tightly controlled by the operator.

1.1 VIDEO AND THE INTERNET

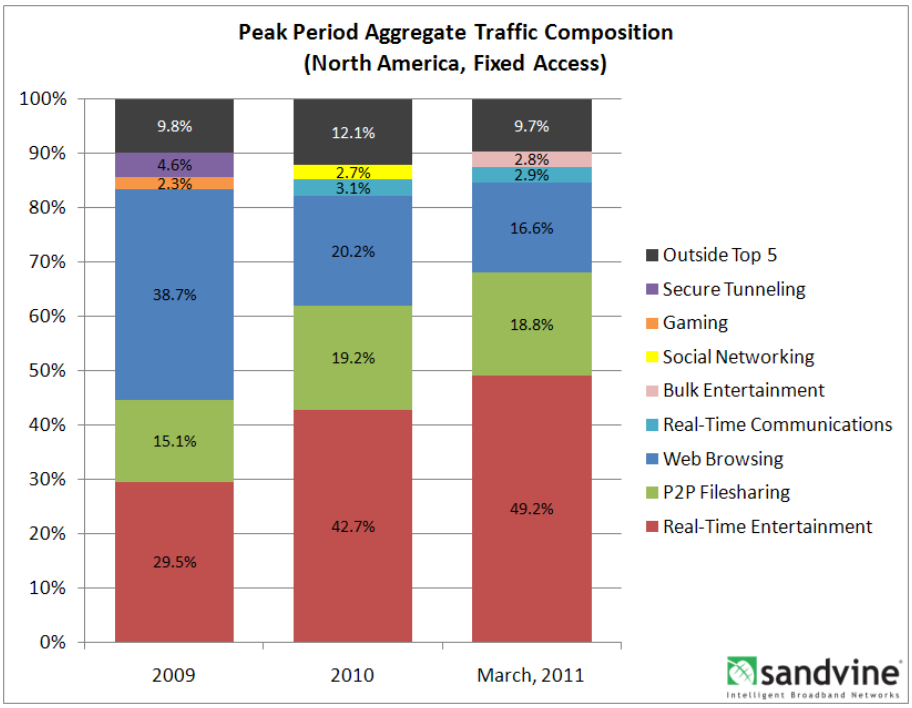


Figure 1.1: Netvine, North American peak traffic (Source: [DeG11; Ros13]).

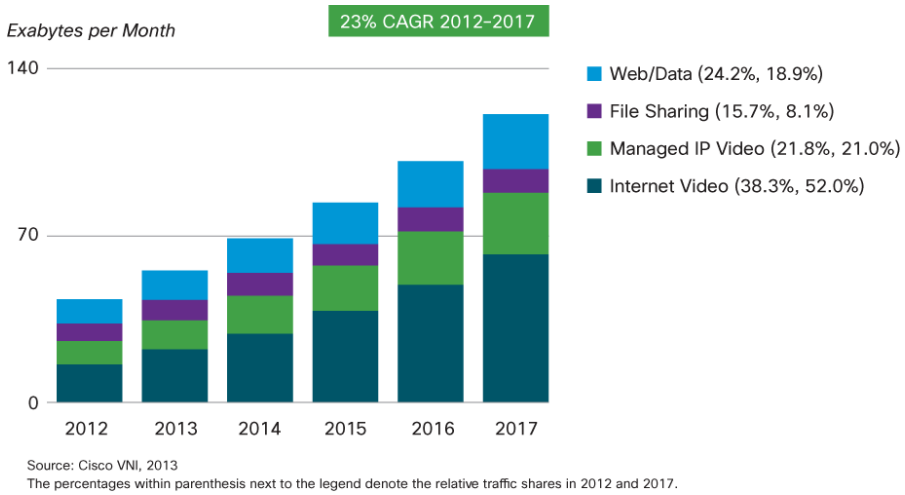


Figure 1.2: Cisco, global consumer Internet traffic prediction (Source: [Cis13]).

The total volume of Internet traffic has been rising exponentially for many years. The composition of today's traffic is manifold, but one of the largest contributing factors is arguably video (take a look at Figures 1.1 and 1.2). Studies predicting the development of the Internet's traffic also tell of the large influence of video in the near future. The two main sources for this video are YouTube[1] and, in countries where its available, Netflix[2], and even newer services like the live-streaming site Twitch[3] acting as runner-up.

Despite not wanting to tune the network to video, it is still very important to understand the dynamics happening around this type of traffic there. With a traffic portion this large any events or behavior specific to this portion will also have a huge impact on the total network. Transporting this video – or rather "streaming" as in watching the video while it's still being transmitted – – integrating themselves much better into the current Web ecosystem – in the Internet has again become a topic of hot discussions. There are long-standing application layer protocols for exactly this use case. Real-time Transport Protocol (RTP), as the most prominent, has been described as early as 1996 [Gro+96]. It is well described in standard educational literature and specifics to it are still being researched and improved on. Interestingly however, it is practically not being used at all in the current updraft of video traffic. [TODO: explain the reason (Web-relationship, tcp basis, firewalls, walled gardens, ...)either here or in a later chapter] Instead, new approaches have been thought out, integrating themselves much better into the current Web ecosystem. And they are using completely different modes of transportation and control. Streaming, and media transport in general, is also not only relevant for the purpose of watching videos, there are many more fields of use with similar requirements. Becoming popular in the last years is the so-called Cloud Gaming: Running virtualized video games on large racks and streaming just the video output to the clients with all the user input handled by the server, putting a huge emphasize on achieving low latency.[4] Investigating the implications coming with these new streaming protocols will be the first task of this thesis.

1.2 THE NETWORK BELOW

Looking back to the actual networks, we also see lots of progress in the past. Despite their rapid evolution, mobile networks, however, still carry much heritage from their circuit switching roots. [5] Starting in the 1950s with early analog predecessors like the German "A-Netz" and first generation cellular structured networks in the eighties, mobile telephony entered the fully digital world with the European GSM and competing Code Division Multiple Access (CDMA)-based technologies around the year 1991. This was similar to the development in the Plain Old Telephone Service (POTS) with its shift away from the analog roots to digital circuit switching technologies like Integrated Services Digital Network (ISDN).

Because of its huge success GSM and its packet-switching extension General Packet Radio System (GPRS) was used as a blueprint for the following mobile network standard

1 https://www.youtube.com
2 https://www.netflix.com
3 http://twitch.tv
4 Select works overviewing this thematic complex are, e.g., [Ros09; WD09; Jar+11; Aus10].
5 For an historical overview refer to https://en.wikipedia.org/wiki/History_of_mobile_phones. [TODO: should remove wikipedia]

evolutions resulting in UMTS (including High Speed Packet Access (HSPA) and High Speed Packet Access Plus (HSPA+)), today's Long Term Evolution (LTE) and even the upcoming LTE Advanced. Through this heritage, many network elements and protocol have hardly changed since the beginning and are still strongly connection orientated with a strong tendency to signaling and statefulness. This creates a wide range of problems which have only begun to show up in recent years due to the large influx of new user and usage scenarios, creating traffic patterns unheard of a few years ago and overwhelming the network's control plane structures.

Traffic in cellular networks follows a development similar to that of the Internet as a whole. Through the advent of affordable high performance smartphones, a thriving mobile application ecosystems, and (relatively) fast access technologies many are now using their phones as the primary device for interacting with the Internet. Video, too, has here become one of the largest contributors of cellular traffic.

However, heavy traffic on a stateful network poses unique challenges to the performance, the design, and dimensioning of network protocols. This gets even more complicated by the circumstance, that operators and vendors usually regard any details on mobile networks equipment as closely kept trade secrets Thus, little is known of the exact make-up of these networks as they are closely guarded secrets of the operator.

This presents us with the second task of this thesis.

1.3 RESEARCH METHODS AND APPROACHES

As is the case with any scientific endeavor, this work is based on a specific set of tools and knowledge of many years of prior research. Measuring, evaluating and modeling is standard practice in many fields. Therefore, many tools are available to tackle problems, but one still has to choose the most fitting ones for a given problem.

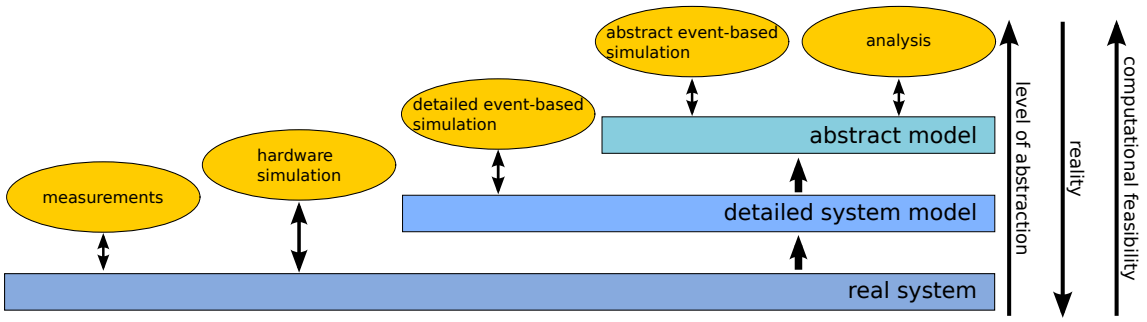


Figure 1.3: Methodical solution spaces and apparatus comparison.

Figure 1.3 attempts to categorize all tools available to a performance analyst, with their corresponding level of detail and abstraction.

The most precise results will always be achieved through actual measurements of the actual system under scrutiny. This will give a point of reference and can be used to validate the accuracy of other methods. But it also comes with a hefty price of huge amounts of data to process and understand. With such measurements at hand, one can now start to make sense of this data and extract the relevant features observed in this process.

Aside from measurements on implementations there are three further possible approaches to widen the scope: emulation, simulation, and mathematical analysis or analytical models. An emulation tries to resemble implemented functionality as closely as possible with the cost of high complexity. While there are many forms of emulation, relevant to our research interests is just the so-called network emulator. Instead of having a real network, parts of it are substituted with emulating hardware or software that treats transmissions according to properties from the actual network. Most typically these properties are the transmission bandwidth, loss, and latency (and variation over time thereof). But emulations always just replace parts of the actual system and therefore its capability to scale is limited.

This is where simulations come into play, implementing all internal and external functionality, including the physical nodes and the network in software. A typical Discrete Event Simulation (DES) can have subtle functional differences but can be scaled almost indefinitely limited only by the available processing time.

Prior to any simulation or emulation and following every measurement on a system is of course the mathematical analysis of existing data. Through this models are built abstracting the system the data is gathered from. With every iteration the model can be refined and new hypotheses formed. A mathematical analysis, for example using queuing theory and stochastic models, can then further broaden the understanding of the system.

[TODO: mention here? Application of queuing theory, modeling the system through arrival processes, service time distributions and a number of servers and waiting places.]

1.4 RESEARCH QUESTIONS AND LINES OF RESEARCH AND GOALS

With these tools at hand, this thesis will follow in most parts two mostly independent lines of research.

The first line will be a thorough investigation of current video streaming mechanisms and services facilitating them. All of the investigated streaming "protocols" are comparatively young and similar.

Protocol is put in parentheses for a reason here. Many of the approaches are not protocols in the classical sense. They do not follow a specification of any standards body, be it International Organization for Standardization (ISO), International Telecommunication Union (ITU) or the Internet Engineering Task Force (IETF). Neither do they show much distinction in the way the use the network and transport video as they simply rely on normal Web protocols. Rather, much of their differences stems just from the behavior of the actual application. The way how it requests and retrieve the data and displays the video, including its reaction to unwanted events, e.g. having insufficient data in the playback buffer. Surprisingly, even just with these rather simple rules, applications can still distinguish themselves from one another. Making the right decisions to events can greatly influence the quality of the video playback.

During the investigation we created a model describing all the common elements in the streaming process. With such a model at hand and in knowledge of the structure and similarities of the applications, one can do many things. Through a performance analysis they can be compared to each other. Interrelations to the network's transport layer and other influences of the network on streaming could be uncovered. This especially includes mobile networks where many assumptions made for wired networks do

not hold anymore. One quick example is the circumstance, that the network layer and all layers above generally do not experience any packet loss in a cellular network. Any loss is treated by the mobile layers, albeit with other side effects on the upper layers. For example, they will experience intermittent times of extremely high latency. This leads to the question, on which kind of information from the network a streaming application could and should depend, or even if there are any actual requirements for streaming.

Gathering this kind of information can result in a better understanding of the quantitative attributes related to these new forms of streaming. The results can be used as an analytical and testing tool for improving the protocols or even lead to different approaches. Furthermore, the thesis aims to provide methods for helping all parties involved in media streaming decide which protocols and methods to choose and which are best suited for specific scenarios.

The second line of research will deal mostly with the described oddities of cellular networks. Through exploration of currently existing networks and architectural specifications data is gathered. This data allows to draw some very interesting conclusion regarding states and signaling in these cellular nets and possible load caused by it. Those conclusions and further evaluations also lead to a model for the core network, abstractly describing the control plane. This will ultimately help to improve the planning and dimensioning of networks, even influence future specification iterations to reduce their fixation on signaling.

For the third and final field of research an attempt is made to bridge both of these lines. With the increasing prevalence of video streaming in mobile networks, it is important to understand the relationship between these two. Here, this will follow mostly a theoretical approach, with descriptions to protocol drafts dealing with these kinds of issues.

Measuring mobile networks is always rather complicated, with a vast number of variables to keep track of and control. To give an example, just think of the movement patterns of mobile devices and the impact this has on handovers between cells as well as the signal strength, and therefore also on packet loss, and, e.g., the video quality. To improve this situation, two methods are explored. Reducing the number of variables – but thus also the level of detail – can be easily achieved using network simulation or emulation. This has also the added advantage of having easy access to all nodes in the (simulated) network. The simulator can then be combined with either traffic from real applications, or the applications under scrutiny can also be simulated. If simulating is out of the question and real-world mobile phone measurements are to be conducted, maybe even as part of a measurement testbed spanning many devices, the phone's state needs to be recorded as precisely as possible. Thankfully, today's mobile phone have a multitude of sensors and information sources available, which will be exploited in the measurement approach present second. Lastly, with the knowledge from all of the investigations, a proposal for a new paradigm managing networking application traffic – especially in cellular networks where mobility matters – is given.

1.5 MAIN CONTRIBUTIONS AND STRUCTURE

Following these research lines, several contributions are made, which will be broken down into three major chapters.

To introduce to the video streaming investigations, covered in Chapter 2, at first the protocols and techniques, that have been used in the past, will be described in Section 2.1. This then leads, to a broad survey of current protocols. A subset of them will serve as the basis for the presented Transmission Control Protocol (TCP) streaming model. Furthermore, to be able to work with the model, an evaluation metric is required, one will be created in Section 2.3.1, as the selection of existing metrics does not fulfill the needs of our model. Now that both metric and model are ready, a measurement study using an emulation approach is conducted in Section 2.4, investigating the performance of an actual video streaming service with the help of the model under the influence of varying network conditions.

The facets of mobile networks, and especially the core network, are researched in Chapter 3. As the amount of specifications as well as architecture, procedure, and protocol details is vast, the mobile architecture background section 3.1 attempts to put all the basics together.

The tackled protocols, systems, and mechanisms are described in Section X. Section Y details the methods that are and are planned to be used for the research. The final section gives a rough estimation on the thesis' schedule. The remainder of the chapter uses data from a passive measurement campaign at a mobile operator. Section 3.3 describes how the data was acquired and how this data can be interpreted. With this foundation laid out, the actual goal of the investigations will now be defined in Section 3.4: an alternative definition and investigation of load in a mobile network, based the control plane and not just the user plane. This goal is pursued throughout the exploratory evaluations of the dataset in Section 3.6. Learning from the data novel load models are then formulated in 3.8 and thoroughly tested using a queuing simulation. With these models mobile network operators can better dimension and plan their networks according to control plane load and not just to accommodate user traffic.

Finally, some thoughts and approaches to study and improve traffic – especially video streaming traffic – in mobile networks are collected in Chapter 4. As mobile devices depend on a lot more variables (with mobility being the most prominent) than devices with fixed Internet access, any measurement needs to keep track of these. Section 4.3.2 presents our approach to tackling this. Section 4.3.3 adapts the earlier described streaming measurement model to work with a mobile network simulation and emulation. And finally, Section 4.2.2 presents a model to improve any traffic that can be broken down into segments, .e.g. adaptive streaming or Web traffic, by providing cross-layer Application Programming Interfaces (APIs). [TODO: add the chapter specifics, when they are fleshed out]

2

MODELING AND MEASURING RELIABLE VIDEO STREAMING

The Web wouldn't have seen that big an increase in popularity and traffic if it weren't for the tight integration of video streaming into every browser. Most forms of today's Web-based video delivery take advantage of HyperText Transfer Protocol (HTTP) and TCP to transport video. This is a completely different approach to what was used before and is traditionally understood as video streaming. Streaming itself can be conducted in many ways, resulting in an ever-increasing number of protocols. Furthermore, the current boom in smartphones creates an increasing plurality of access network technologies. Any network shows characteristic properties, or Quality of Service (QoS) metrics. These are typically:

- The *bandwidth*, or the maximum throughput a user can achieve, which is always limited by at least one link, serving as the bottleneck. In most cases this will be the access link, but can also be any other link in times of high load. Bandwidth on an access medium can also be shared between all of its participants as is the case with any radio technology or cable Internet access.
- The *delay* is the time data travels between a sender and a recipient. The term *jitter* is used for the delay variation and occurs for example when successive packets travel on different routes or through different radio receivers during mobility.
- *Loss* occurs when data packets do not reach the target. An imperfect physical medium can flip some bits in a packet, which the responsible protocol layer will recognize and drop the packet or re-request it.

Video streaming needs to cope with all of these circumstances and still work well. This chapter investigates the model Web streaming uses. It differs significantly from models for traditional streaming, which are mostly specific to a single protocol. The presented method rather aims to evaluate the performance by capturing generic behavioral patterns of streaming mechanisms from the perspective of a streaming application. Specifically, the model is based on the level of the playback buffer, which is a common attribute to any media playback. Thus, it can consider any network and playback behavior, and maintains flexibility with regards to the actual streaming server implementation, type of transmission network, protocol stack, and codec.

After defining an appropriate performance metric, this model is implemented in a network emulation testbed. A measurement campaign on YouTube is then conducted while the network is being subjected to emulated QoS parameters (compare also our with publications [Met+11] and [MRT12]).

2.1 BACKGROUND

Before diving into the model some technical groundwork has to be laid. We describe protocols commonly used in the past and present and how to classify them. This is followed by other work related to this approach.

2.1.1 *Definition*

Any digitally stored video consists of a number of frames, organized into variable-sized groups, and audio samples which are played in sequence. Frames, single images of the video, do not only make use of typical spatial image compression mechanisms but encompass also temporal motion compensation, creating a dependence between frames. Videos can be encoded with a bit rate that is constant or variable over time. Typically, a variable bit rate encoding is chosen as these schemes offer a higher compression rate. To correctly display a frame, all previous frames in a block need to be present.

Streaming, or to be more precise video streaming, is the process of playing a video while it is still being transmitted over a medium. As there is no need to have a file stored locally, received frames are typically put in a buffer to be played at the correct time. The amount of buffered video depends on the allocated buffer size as well as the video bit rate, and the transmission bit rate. It can also be controlled by the time offset between receiving the first frame of a video and actually playing it.

2.1.2 *Streaming Classification*

Video streaming is a broad term covering a wide spectrum of applications as well as possible implementations. To break down and classify this field we define the following criteria to make a distinction.

VIDEO SOURCE The first criterion is the source of the video with the two major sources being a file stored on a remote server or a live source. Stored video can be streamed and played at any point in time. Live sources, on the other hand, are transmitting only at a fixed point in time. Depending on the type of content the timeliness of playback may also be important (imagine you are watching a game that is played right now).

ADAPTIVITY OF CONTENT Video streaming can also be distinguished based on its adaptivity. In the simplest case there is no adaptivity present and the video is available in only one bit rate (which may still be a variable bit rate). But there are cases where an adaptation of the bit rate would be helpful. For example to accommodate for a clients needs in matters of the screen size. Usually, adaptation is used to tune the video stream to the currently available connection bandwidth. Adaptation can be achieved in two ways. Either to prepare and store several encoding levels beforehand or by encoding on-the-fly to a specific target. While the latter approach can adapt much more specific to certain goals it cannot be precomputed and will require more compute time when the number of clients becomes larger.

Adaptation also increases the amount of necessary control and information exchange. In the simplest case, streaming would only require a single command to start the streaming while any single adaptation adds another set of commands.

LOCATION OF CONTROL Another matter is the location of control for a stream, with several possible ways to choose from. We distinguish between horizontal and vertical control.

In a horizontal direction control can be placed either at the streaming server, the streaming client or possibly somewhere in the network path in between.

A controller at the client typically just means that the video player itself is in control of the streaming process. The player starts the streaming and adjusts its requests to the server based on the player's needs. In this situation the server can be very lightweight as no decision logic needs to be present there. This is also called **pull-based** streaming.

Control can also be placed at the server with a stronger emphasis on the information available at the server side, making it easier to coordinate and adapt to a larger number of streaming clients. Similarly, this is called a **push-based** approach as video data is pushed to the recipient. For control to work properly state has to be kept imposing a certain memory overhead. This can become significant and a limiting factor for large streaming servers. Contrary, pull-based streaming usually does not require much or any state at all at the server.

Control information may need to be exchanged to communicate the state between the two endpoints. This can happen either explicitly through the exchange of signaling messages, or implicitly by drawing conclusions on another participants resources and behavior, for example through other protocols in the stack.

While not being able to control everything about streaming, the network may still be able to influence or manipulate an ongoing video stream. (Non-)Transparent proxies come to mind, which could intercept streaming requests and redirect them to another server located in the proximity of the requesting client. A network can explicitly expose network's quality of service data to applications or these application can make reservation requests to the network.

Additionally control can be distributed vertically at different positions in the protocol stack. While usually streaming is conducted through a dedicated application layer protocol or even directly through an applications behavior, portions of control functions can also be offloaded to deeper layers. A typical example would be the use of TCP for reliable streaming.

RELIABILITY OF UNDERLYING TRANSPORT PROTOCOL A major differentiation can also be made based on the reliability of streaming. Streaming can either act similar to a simple file download and just progressively download the video file in question while already playing it. This is conducted by using TCP as a transport protocol, guaranteeing that no packet is lost in the process. TCP does this by retransmitting packets it thinks are lost with the price of added latency and reduced throughput during retransmission. This reliability can however also cause the progress of the whole video stream to stall, If video data does not reach the client in time before its playback buffer is depleted, and therefore a perceptible loss of quality. This situation can be alleviated or even avoided by carefully planning the playback process and the buffering behavior.

On the other side stand streaming protocols that base themselves on User Datagram Protocol (UDP), which offers no reliability features as TCP and just sends out packets as-is. When packets are lost, the video can still progress but parts of the video output may be distorted or lost. Additionally, unreliable streaming protocols must take over other control features, that would otherwise have been taken care of TCP. The adherence to an alloted or fair share bandwidth and congestion control come to mind, or else a high usage of this protocol could again lead to another congestion collapse [Nag84].

Transport protocols that offers congestion control but no reliable delivery might be a desirable middle ground between these two extremes. Datagram Congestion Control Protocol (DCCP) [KHF06b] is an example for such a compromise and might prove beneficial for the streaming process.

MULTIPLEXING OF DELIVERY Finally, the number of targets of a single video stream can also differ. A stream is unicast if the control loop is exactly between one sender and one recipient. Servers can still support multiple unicast streams at once, they are just completely independent of each other. A multicast, or even a broadcast, stream is simultaneously sent to a group of recipients, stream control is established at the sender for the whole group. Therefore, multicasting is always using a push-based approach to control.

2.1.3 *Survey of Protocols*

With these classification criteria at hand, we can now start looking at actual protocols, and find out, which motifs they are following. The section largely describe RTP and compare it with HTTP-based approaches including Dynamic Adaptive Streaming over HTTP (DASH) while also mentioning some other, proprietary, streaming protocols.

2.1.3.1 *RTP and Related Protocols*

RTP [Sch+03] is always used in conjunction with its sister-protocol RTP Control Protocol (RTCP) and often also employs Real Time Streaming Protocol (RTSP) [SRL98]. According to literature, they are the classic approach to video streaming (for example compare [KRA08, p. 589ff] and [PD07, p. 426ff]). The protocol suite employs a *push-based approach*, the RTP server application has full control of the streaming process. Control and information exchange is also out of band through RTSP and RTCP. Therefore, multicast is also easily possible with RTP but not mandatory.

RTP has also no inherent adaptivity nor reliability mechanisms, neither does it conduct congestion control on its own. Moreover, RTP generally runs on top of UDP, which also does not provide congestion control. All must be provided by the server-side application implementation if necessary. In case of multicasting the potential to conduct transport adaptations is very limited, as the server has to take all the recipients into consideration for its decisions.

RTP RTP itself provides just the packet format and header for the transport of the actual multimedia data. Any stream type is transported in a separate session. This includes the presence of both video and audio, which must then be synchronized to each other. Each session uses its own UDP source-destination port pair.

The RTP specification itself defines only the most basic packet header, with several additional specs describing dedicated profiles for various content types. For today's prevalent MPEG-4 protocols, including H.264, multiple profiles, defined in [Mee+03; Wan+11a; Sch+11] , and with this many ways to embed video into RTP packets, are available. Common to all is the variable-size RTP header of at least 16 B. Video codecs may embed their own organizational structure inside the packet. For example, if dealing

with an MPEG-4 Elementary Stream (ES), the payload may contain one or more Access Unit (AU).

RTCP RTCP is used to exchange feedback and control information between receivers and sender and vice versa. These sender and receiver reports are transmitted on a separate UDP connection at small intervals scaled in such a way, that the bandwidth should not exceed 5% of the stream's bandwidth. The reports will include statistics related to lost packets and the packet delay and variation. Based on these, a sender can adjust its streams to fit the current conditions. Likewise, a receiver may tune its video buffering behavior or may even switch stream sources.

STREAM INITIATION RTP/RTCP itself provides no means to discover, initiate, and control the streaming process and has to rely on additional protocols. RTSP is on of these, sitting atop of either UDP or TCP. It provides a set of commands the client can issue to a streaming server to control a stream and the streaming state at the server.

In case of multicasting, stream management can also be conducted directly by joining predetermined multicast groups through the use of Internet Group Management Protocol (IGMP) [HCH06] without the need for RTSP. This requires the cooperation of all intermediary routers. Therefore, it is usually only seen in closed networks ("walled gardens"), where the whole network infrastructure is owned by a single instance. For example, Telekom Austria's A1TV employs this scheme. [TODO: reference, bernhard?]

RTP is also used extensively in conjunction with a lot of other protocol suites, including Session Description Protocol (SDP) [HJ98] and Session Announcement Protocol (SAP) [HPW00] for stream discovery or in realtime communication protocols such as Session Initiation Protocol (SIP) [Ros+02] and Extensible Messaging and Presence Protocol (XMPP) [Sai11a; Sai11b] with the Jingle extension.

However, the requirement of several open UDP sockets has issues with the presence of middleboxes, especially Network Address Translation (NAT) nodes, because of the difficulty to forward incoming UDP packets to the destined host. This can be, sometimes though unreliably, circumvented by using NAT traversal techniques like Session Traversal Utilities for NAT (STUN) [Ros+08] or Interactive Connectivity Establishment (ICE) [Ros10].

2.1.3.2 *HTTP Streaming*

When compared to RTP, streaming based on HTTP uses a much less intricate approach and only reuses existing protocols. HTTP/1.1 [Fie+99] is the basis of the Web and is a request/response protocol mainly to retrieve and pull files from and to a remote location. The protocol is stateless for the server, any request is treated as standalone and will be responded to only with the provided metadata.[1] This holds true even when more than one request is sent over the same TCP connection, which can be done with persistent HTTP connections. Additionally, requests can be sent over one connection without waiting for the answer of the previous request. This is called pipelining and can reduce the round-trip time delay between two consecutive requests.

But HTTP can also be easily exploited for media streaming. The file to be retrieved should of course contain video and all frames have to be stored sequentially. If there

1 State can still be achieved through other paths, like cookies, but this is out of scope.

are separate streams present in file, most commonly at least video and audio, they must be interwoven. Necessary video metadata, which includes information on the codec and the streams, needs to be at the beginning of the file or at least before the position in the file where its needed. Alternatively, streams can also be stored in separate files, potentially simplifying the file structure. However, this increases the complexity of synchronizing both streams at the video player.

The actual streaming is controlled completely by the player application at the client. This player simply has to issue a HTTP 'GET'-request to a video file located at a Web server. The file can already be read during the transmission process and extracted video data will be put in the player's buffer. If there is enough video in the buffer, playback can be started. The complexity in this process comes from the need to keep track of the amount of video in the buffer and avoid to it run out at any point during playback. Approaches to this task will be explained in detail in Section 2.3. HTTP also allows so-called Range Requests, which allows to download only certain portions of a file, indicated by the Byte position. Streaming players can exploit this to enable skipping to certain positions. This again needs metadata to correctly infer the byte position in a file from the video playback position. Else, the Range Requests have to be guessed.

RELIABILITY AND ADAPTIVITY HTTP uses TCP as transport protocol, which has implications to HTTP streaming and makes it so distinct from RTP/UDP-based approaches. TCP's three large features are arguably reliability, congestion control, and flow control.

Reliability means, that at the transport layer and above no packets are lost and file requested by a HTTP application will always be transmitted in full to the client (as long as the connection is not completely interrupted). TCP's sender side detects lost packets either by timeouts waiting for the corresponding acknowledgment or, preferably, through duplicate acknowledgments of previous packets. If either of this happens, the lost packet is retransmitted, causing a noticeably increase and variation in latency. But this also means, that the transmission of all consecutive packets has to wait on this one. On links with high loss the transmission can be stalled to such a degree that the incoming bitrate is lower than the bitrate of the playing stream, draining the buffer until it runs empty. Keep in mind, that with reliable streaming, frames, or parts of it, cannot be dropped, and the whole video will always be played.

In addition, TCP also employs congestion control and avoidance mechanisms. While the sending rate of an UDP application is completely controlled by the application's logic, TCP detects and throttles the transmission to its fair share of the current connection. This can also cause the transmission rate to become lower than the video bitrate. The third transmission rate influencing mechanism is flow control. The receiver (and especially also the receiving application) can notify the sender, how much data it can receive in the next time window and thus can throttle the transmission rate itself.

This is an important method of control for the player. Usually, TCP's transmission fair share rate is expected to be much higher than the stream's video bitrate. While this makes sense for a simple file download to finish it as soon as possible, this behavior is unwanted for streaming. Rather, one wants to match the stream bitrate, with a bit of additional headroom to compensate for rate variations, to keep the playback buffer size in certain bounds that neither overwhelm the receiving device nor should the buffer be in danger of running empty.

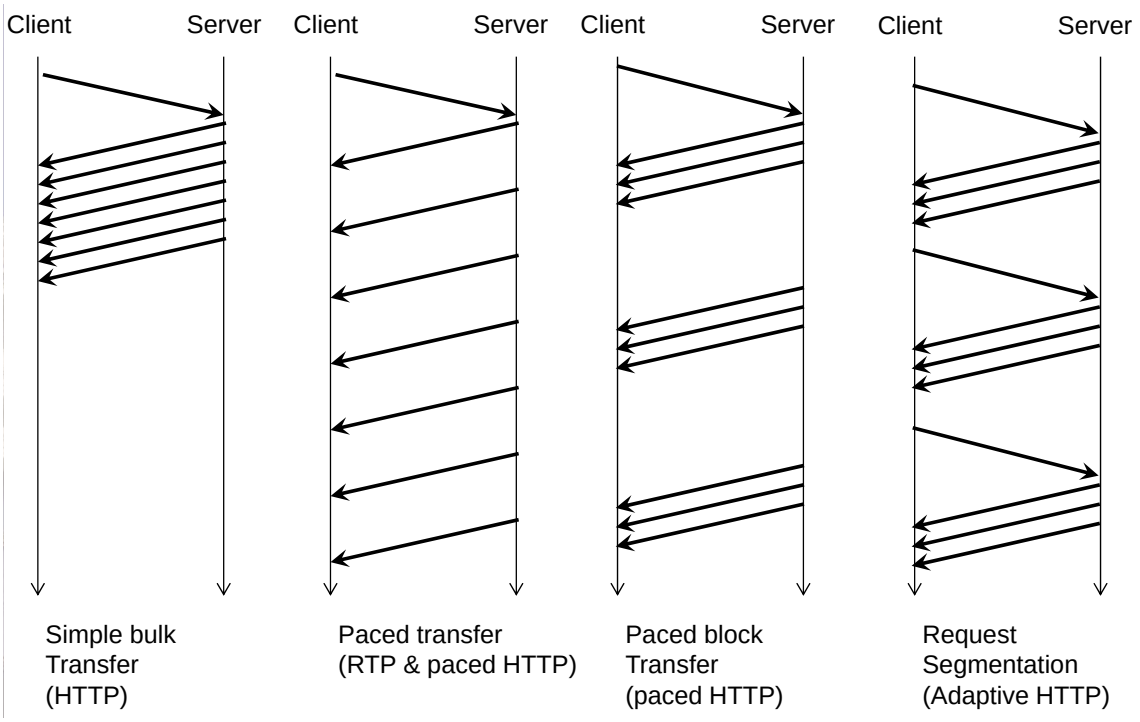


Figure 2.1: Comparison of several possible streaming transfer modes [Ma+11].

[TODO: reference to transmission pattern here? alternatively, put one of the old youtube throttling pictures here]

The first of the alternatives to achieve control over the playback buffer using HTTP-streaming is to appropriately size TCP's flow control receive window by the application. Alternatively, the HTTP-server can also manually throttle the download process, through various pacing strategies. The second and third transmission diagrams in Figure 2.1 depict two possible strategies compared to a regular HTTP file transmission in the first transmission diagram. A third way, is to either partition the stream file into smaller even-spaced segments, that have to be requested independently, or use the aforementioned range requests on the stream's file. Through this, the receiving application can delay the request of new ranges or segments so that it matches a targeted bitrate over a longer timeframe.

These mechanism, however, can also result in a very bursty block-like transmission, a so-called ON-OFF pattern, and can cause undesirably interactions with TCP's flow and congestion control mechanisms [TODO: ref to interaction]. Overall, stretching out the transmission may reduce server load spikes and the required buffer size on the client device but also makes streaming more vulnerable to insufficient network QoS parameters. These specific approaches to pace to a target rate can generally be subsumed under the term *Application Layer Flow Control*, which is also being implemented by some Web streaming services, e.g. YouTube [AN11; Met+11].

The aforementioned only adapt the transmission rate to the stream's bit rate and not the stream rate itself. Different video bitrates may be desired for many use cases, especially reacting to changing network conditions, take vertical handover from a 802.11 to a UMTS network with a much lower throughput as an example. Quality adaptation with HTTP streaming is generally achieved through the described range request or file

segmentation mechanisms. For both approaches, multiple versions of the file or the segments have to be generated in different encoding quality levels. Also, the video file format needs to be able to support switching the stream and have an index to correlate the video files with their quality level and temporal position. [Ma+11; BAB11a]

Several formal adaptive streaming protocols are in the process of standardization. HTTP Live Streaming (HLS) [PM13] defines a playlist format to be stored separately on a server that links to all available stream variants and segments thereof in sequence. DASH [Sto11] is a ISO/International Electrotechnical Commission (IEC) [MPE12] and 3GP-DASH [3GP13l] standard. Herein, all video segments are gathered in an alternative XML-based presentation scheme. Due to its file-based, using HTTP for streaming is much more suited for stored video. But has also been successfully employed for live content, both HLS as well as DASH support this.

Other than the two streaming endpoints, the network can also, to a degree, control parts of HTTP streaming. HTTP allows to have forward and reverse proxies placed in the transmission's path. These could potentially alter and adapt the stream, which is however usually not done. Through Domain Name System (DNS) video stream requests can also be redirected to a server instance chosen by the stream source. The content provider has to internally distribute the stream files to all the caches, that are advertised through DNS. Caches are usually placed in close vicinity of potential receivers to avoid This creates a so-called Content Distribution Network (CDN) avoiding long duplicate transmission paths through the Internet and the bottleneck at a single server. CDNs can also be used to achieve a multicast-like effect, which TCP-based streaming cannot provide itself. Only traffic on very few links close to the recipient has to be replicated. However, the the access links of stream receivers are typically separate entities anyway, so even multicast enabeld streaming does not save bandwidth there. For an investigation of YouTube's CDN structure refere to [Raf+11].

Currently, HTTP is in the process of receiving major remodeling with the efforts of WebSocket[2] [FM11], SPDY [BP11; BP12], and ultimately also HTTP/2.0 [Bel+13]. All three improve the multiplexing capabilities of HTTP and allow the server to initiate transmissions on its own. This enables more control possibilities for the server and ca improve any segment-based adaption scheme as these segments do not need to be requested anymore but could just be pushed by the server at a convenient time.

2.1.3.3 *Other Approaches and Classification Matrix*

There are also other proprietary and standardized streaming systems usually tailored to specific requirements and applications. Multimedia Broadcast Multicast Services (MBMS) [3GP08e; 3GP08f] is a Third Generation Partnership Project (3GPP) specification for multicasting multimedia traffic in the mobile network architecture. The explicit control structure of protocol suites like MBMS and also IP Multimedia Subsystem (IMS) [3GP08c] weaves application and network layer tightly together. This theoretically allows for an improved streaming performance at the cost of universally applicable behavior. Real Time Messaging Protocol (RTMP) [PT12] is a proprietary streaming protocol which in the past has seen widespread use through its implementation in the Adobe Flash Player and Plugin in Web Browsers. Also leading a niche existence are Peer-to-Peer (P2P) based streaming approaches. In P2P there is no explicit server. Instead,

2 http://www.websocket.org/

connections are made and stream data is exchanged between equal hosts, avoiding a centralized server's bottleneck. P2P streaming is used for example, in Tribler[3] and Zattoo[4].

Table 2.1 attempts to summarize all the protocols of interest to this research and apply the proposed classification criteria to them.

Table 2.1: Protocol Classification Matrix.

Protocol	**Vertical Location of Control**	**Horizontal Location of Control**	**Reliable Transport**	**Video Type**	**Adaptivity**	**Multicast**
RTP	out-of-band application layer protocol	server-side and limited intermediary (translators and mixers)	unreliable (UDP)	low delay live streaming	server-side adaptation (transcoding)	yes, IGMP
simple HTTP	in-band streaming application	client-side	reliable (TCP)	stored, not live	none	no, but emulated through CDN
adaptive HTTP (DASH)	in-band streaming application	client-side	reliable (TCP)	stored and near-live	client-side with file segmentation	no, but emulated through CDN

2.2 RELATED WORK

Video streaming touches many aspects of computer communication network research. The technical fundamentals for video streaming have existed for a sufficiently long time so that there is a large body of existing work. The presented research

The chapter's focus lies on reliable HTTP streaming to which [BAB11a; BAB11b] and [Ma+11] give an introduction and overview the mechanics involved in streaming, e.g. flow control mechanisms in the video delivery. Akhshabi et al. [ABD11] take a look at real world streaming implementations and conduct comparative measurements. Initial experiments evaluate the viability of this kind of approach.

Concerning specific streaming solutions, there are several publications discussing YouTube's architecture. In 2007 Gill et al. [Gil+07] made a long-term observation of YouTube traffic originating from an university network. Their analysis shows detailed

3 https://www.tribler.org/trac
4 http://zattoo.com/int/

characteristics of the served videos, among others file sizes, durations, and bitrates, and reveals a daily number of video requests. Adhikari et al. [AJZ10] collected data from points of presence of one ISP to explore the service's traffic distribution and load balancing techniques. However, these were still based on YouTube's old architecture prior to being acquired by Google. The new infrastructure differs largely from the old, e.g. load balancing and content distribution now exclusively uses Google's network. Mori et al. describe in [Mor+10] distinctive attributes of video traffic flows originating from YouTube's new and current setup while Torres et al. [Tor+11] focus on observations of the CDN's server selection process using multi-network passive measurements. Concerning CDNs, [Lab+10] showed the importance of this new traffic distribution approach in interdomain traffic. Web and video traffic was also on the rise, both is to be expected through the presence of large video distributing Web sites.

The authors of [Wan+03] propose an analytical model for TCP-based video streaming, differentiating between live and pre-recorded videos. The impact of packet loss on an unreliable video stream is studied in [CLC10]. The loss-hiding properties of reliable streaming makes this study only somewhat applicable to HTTP streaming. In [Kaw+10], the authors present a quality-assessment model for video streaming services, with the quality features derived from the actual video. The model does not include the network behavior, but focuses on the codec performance instead.

To determine the quality of the streaming process as well as any models resembling the process, metrics have to be used or the so-called Quality of Experience (QoE), either subjectively or objectively, measured in some way. While an overview will be given in Section 2.3.1, a few selected publications are already presented here. A metric termed "application comfort" defined and applied to YouTube videos in [Sta+10] to monitor live network conditions in realtime. While this approach is in effect similar to our evaluations, it is geared towards a very specific implementation of streaming, whereas our presented methodology is more generic. YouTube's QoE is not just up to the sender and receiver. Some control could also be placed in an access network, manipulating YouTube streams to improve streaming quality. This was conducted in [Sta+11] for a wireless mesh network. In 2012, a publication [SHC12] presented approaches to derive YouTube's playback buffer and quality from passive measurements inside the network. Approaches like this can be used by Internet Service Provider (ISP) to check their network and estimate quality customers are achieving. A publication by Hoßfeld et al. [Hos+11] identifies QoE influencing factors on YouTube streaming through subjective Mean Opinion Score (MOS) collected by a crowdsourcing method. The number of stalling events is seen as the factor with the highest impact. This idea is furthered by [Hos+12] with a comparison between the initial day of a stream's start and the number of interruptions, with stalling resulting in a much lower MOS.

Observations performed in [Ket+10] on the Android platform were showing that re-buffering events, i.e. phases of video stalling, can result in a large drop in MOS. The authors of [MCC11] measure QoE effects of HTTP video streaming in a controlled test setup and also conclude that degraded network QoS increases re-buffering frequency and decreases the MOS. Gustafsson et al. [GHP08] investigate the loss in perceived streaming quality and establish a parametric objective opinion model. [SHR12] presents another QoE model that attempts to estimate the quality of adaptive streaming with neural network trained through subjective tests.

Adaptive streaming has also been a topic of intense research. [DMP11] investigates quality adaptation techniques and proposes a feedback mechanism for quality control. Unfortunately, this places control solely at the server side, contrary to the current trend of the streaming client exercising full control. The authors of [CM10] conducted measurements of yet another server-controlled adaptive streaming mechanism, which is employed by Akamai's video streaming.Moreover, according to [HHM11], the circumstance, that TCP throughput does not automatically throttle itself to the current video bitrate, could be problematic. The paper proposes and tests an additional scaling mechanism in a simulation.

To get a grip on the real-world behavior of streaming mechanisms, many measurement studies are conducted. In measurements, especially in passive measurements, one typically cannot measure the application protocol on its own. Rather, the whole network stack, starting from IP packets and upwards is captured and must be evaluated for the specific property under investigation. In [Erm+11] such a general traffic study is conducted in a cellular network, estimating the ubiquity of video streaming, even in mobile networks. The authors of [Hua+12] conclude in their proxy-based active measurements that many adaptive streaming approaches underutilize the available network bandwidth and achieve a lower quality than they could. An analytical ON-OFF model is developed in [Rao+11] through the evaluation of active measurements comparing different streaming strategies. In a survey of several streaming protocols [MHM10] the overhead on the transmission of each of them is investigated and compared based on an analytical approach.

2.3 STREAMING MODELING

As stated, the goal of this chapter is to model the measuring process and reliable streaming. This section will introduce our reliable TCP-based streaming model. It is intended for easily comparing different protocol variants against each other and measuring all variants in one testbed.

But first, because any evaluation of a model requires metrics, we describe ones appropriate to the model and give a rationale.

2.3.1 *Metrics for Reliable Transport Streaming*

When measuring anything related to video or even just image quality, one has the choice between conducting a subjective or objective assessment.

During a subjective test, human assessors evaluate and rate video quality in a controlled environment with the results usually being aggregated into an overall relative quality score MOS. Through the human element, conducting a subjective assessment is very time and resource consuming but it also achieves the highest precision.

This is where, objective video quality assessments come into play. Modern objective models attempt to recreate the features of the humans' visual perception and psychophysics and are calibrated by subjective assessments. Most objective models operate on a full reference approach, they directly compare the original reference video to the resulting video after being encoding or transmitted.

Two of the simplest full reference image quality metrics, which can also be applied on video, are the Mean Squared Error (MSE) and Peak Signal-to-Noise Ratio (PSNR), defined as:

$$\begin{aligned} \mathrm{MSE} &= \frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2 \\ \text{and } \mathrm{PSNR} &= 10\log_{10}\frac{L^2}{\mathrm{MSE}}, \end{aligned} \tag{2.1}$$

with N as the number of pixels in a frame and individual pixels x and y from the reference and output frame respectively. L denotes the maximum value of a pixel. For grayscale or when investigating each color channel separately, usually $L = 8\,\mathrm{bit} = 255$. [WSB03]

Image quality models can by nature only test for spatial distortions of a single image. This includes a general blockiness or blurriness, noise, or reduced resolution. Dedicated video quality assessments can additionally take temporal metrics into account, e.g. frame rate anomalies. Such models are being researched and standardized by the ITU and Video Quality Experts Group (VQEG) for example in [ITU04; ITU08b; ITU08c].

When conducting dedicated streaming quality measurements, only this portion should be taken into account by a metric, and not the initial encoding process. During streaming only a specific subset of quality degradations can occur. Lost or late packets can cause missing blocks in a frame or frames to be skipped completely. Initial thoughts concerning Internet Protocol television (IPTV) QoE have been given in [ITU08a] and the influence of packet delay variations on playout buffers is investigated in [DC02]The Media Delivery Index (MDI) [WC06] is an attempt to capture this behavior and relate it to the network QoS. Its metric relies on two properties, the Delay Factor (DF), as a measure of the network's latency and jitter, and the media loss rate. Of special interest to this investigation is the DF, which is calculated based on a virtual buffer (VB) of received stream data as

$$\begin{aligned} \mathrm{VB} &= r_{rcv} - r_{drain} \\ \mathrm{DF}_i &= \frac{\max(\mathrm{VB}) - \min(\mathrm{VB})}{r_{drain}} \end{aligned} \tag{2.2}$$

Reliable streaming has even less possibilities to degrade a video stream. Packets can not be out of order and loss is concealed by TCP, meaning that the transmitted and the played video are identical. The only thing that can still happen, is portions of the video arriving too late to be played out at their intended point in time. A potential reliable streaming quality assessment metric needs to keep track of the following properties:

- The initial delay, which is the time delta between the start of the transmission and the start of the video play.
- The number and lengths of interruptions or stalls during playback.
- For adaptive streaming, the characteristics of the quality levels the video was played in. This includes the number of switching events and the duration of each level.

A concise metric covering all properties has not been defined yet. The Continuity Index (CI) was defined in [Zha+05] and used to determine quality in P2P live streaming. It is defined as "the number of segments that arrive before or on playback deadlines over the total number of segments" and with this partly captures the stalling property. In [PP11] and [SBP13] a so-called Pause Intensity (PI) is defined and evaluated. The definition $I_p = \mu\nu$ is simply based on the number of stalls μ and the average stall duration ν.

Generally, most research operates just directly on these three properties, which allows for the most freedom in usage scenarios. The streaming measurement model presented in the following section also works with the assumption, that only the three properties are of importance. The model assumes no special metric, the individual properties can be directly attained from the model. Though any metric could still be applied on the results.

2.3.2 *Measurement and Playback Model*

Parts of the model presentation is based on the author's previous work published in [BMT12], [Met+11], and [MRT12]. It is based on the desire to compare all in-the-wild variants of reliable streaming protocols in a simple and concise way. This is achieved by basing the model on the component that is common to all of the approaches: the playback buffer.

To display a video stream, an application needs to maintain a playback buffer of sufficient size to at least gather enough data to reconstruct one single atomic unit of playback such as a video frame. From the perspective of a player application , a video consists of a sequence of atomic units, video frames and audio samples. The application progressively decodes the video from a source and stores the units temporarily in a memory buffer before playing them. In reliable streaming, the buffer is filled by the payload from received TCP segments a subject to the network QoS. The process can be subsumed as:

$$buffer(t) = \sum_{0}^{t} data_{received} - \sum_{0}^{t} data_{played}$$

Both the incoming and outgoing data stream are variable over time. The fill level of the playback buffer is the critical component in the playback process and the central element of the model.If the buffer reaches a size of zero the playback process stops and stalling occurs.

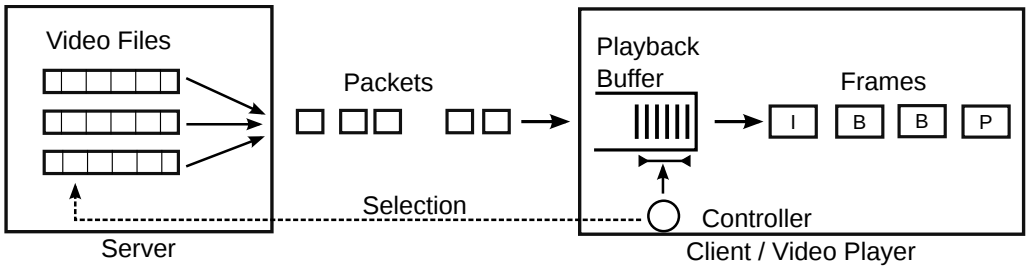


Figure 2.2: Reliable streaming playback model based on buffer control.

Figure 2.2 overviews the reliable streaming model. The controller, part of the video player, selects a video from a remote location and the transmission is started, filling the playback buffer. The model has three degrees of freedom, which are all governed by the controller and together are coined playback strategies. These are:

- The initial playback delay, which is the time between the initiation of the video stream transmission and the actual stream playback. The larger this is chosen, the bigger the safety margin on the buffer gets. If the video and transmission bitrate are known to be constant and and appropriately dimensioned, the initial delay can be chosen to be very small.

- Playback pause and resume decisions based on the current buffer fill level. This is a generalization of the initial playback delay, which is in fact only one, albeit always occurring stalling period.

- Selection of the video or video segment with a video bitrate chosen according to the current network throughput. This is only applicable for adaptive streaming.

These decisions yield a stalling period distribution for a streamed video. The frequency and the duration of stalls directly relate to the decision function of the playback model. The more frequent the stalls are, the shorter they will be; if the strategy produces longer stalls, they will be less frequent assuming the same network conditions. The time scale on which streaming applications buffer content usually lies in the range of seconds. This is a necessity in best-effort networks, as the available network bitrate might drop unexpectedly and cause stalling.

The the rest of this sections present fundamental playback strategies and strategy building blocks with features extracted from real world examples, which are given afterwards.

2.3.2.1 *Null Strategy*

The simplest strategy is having no strategy at all. Playback is started immediately when at least a single frame fully resides within the buffer and stops again at an empty buffer. The behavior can be summarized as "Whenever anything can be played from the buffer, do so".

This results in frequent stops and a large loss in playback continuity and will therefore not be used in practice. However, this strategy has some interesting theoretical properties, which is why it is mentioned here. It minimizes total stalling time and the required buffer space. Moreover, it gives an upper limit for the number of stalls occurring[5]. Therefore, it can act as a baseline reference to assess the performance of other strategies.

Figure 2.3 depicts an exemplary time series diagram of the contents of a video buffer using this strategy with a transmission rate only slightly above the video stream's rate. The buffer frequently drops down to zero forcing a short stall. According to the presented related work on the QoE impact of stalling frequency in comparison to the length of stalls [Hos+11], this is the worst possible scenario for a person watching the stream.

5 As a video frame is atomic, no other model could possibly stop the playback more often.

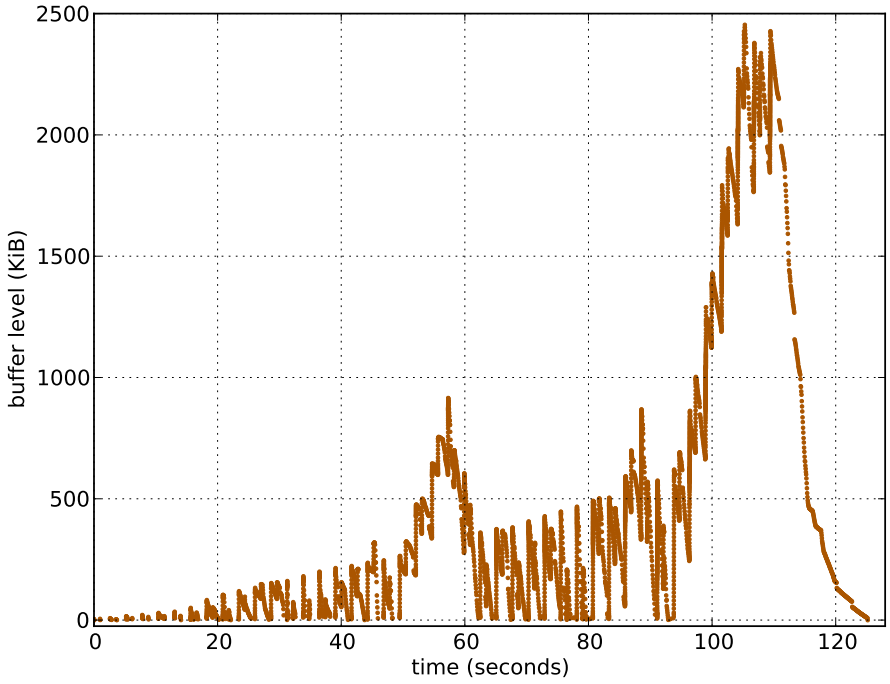


Figure 2.3: Buffer fill level with null strategy; 33 s total stalling.

2.3.2.2 *Threshold Strategies*

Instead of instantly restarting playback, a lower threshold can be introduced. Only after a certain buffer fill level threshold has been surpassed, playback will be started. Thresholds can be set independently for the initial playback delay and stalls, with the initial playback delay generally set to be higher.

The threshold can be chosen in a number of ways. It can either be an absolute data volume, a buffered video duration. The latter is much more suited for variable bitrate videos as it automatically adapts itself to the current bitrate. A third option is to buffer for a certain amount of real time – this can be seen as threshold – and starting playback after that period regardless of the volume of the buffer. Additionally, the threshold could also either be set to a constant value or dynamically chosen according to the expected network QoS.

Besides this single-threshold strategy, a two-threshold strategy might make more sense for segment-based streaming. In addition to the lower threshold, an upper threshold is introduced. When reached, no new segments will be requested until the buffer arrives at the lower threshold again. To achieve an hysteresis effect a third threshold, somewhere between the lower and upper bound, can also be introduced. Through this, the maximum buffer size can also be controlled. This is important in situations with hard limits on available memory. Mobile devices come to mind here.

An example buffer diagram is displayed in Figure 2.4. In this case, the initial delay was controlled by a buffered video duration threshold of 2 s and a resume condition also based on buffered video duration but with a 5 s threshold. The strategy produces noticeable less stalls than the null strategy but slightly increases the total stalling time.

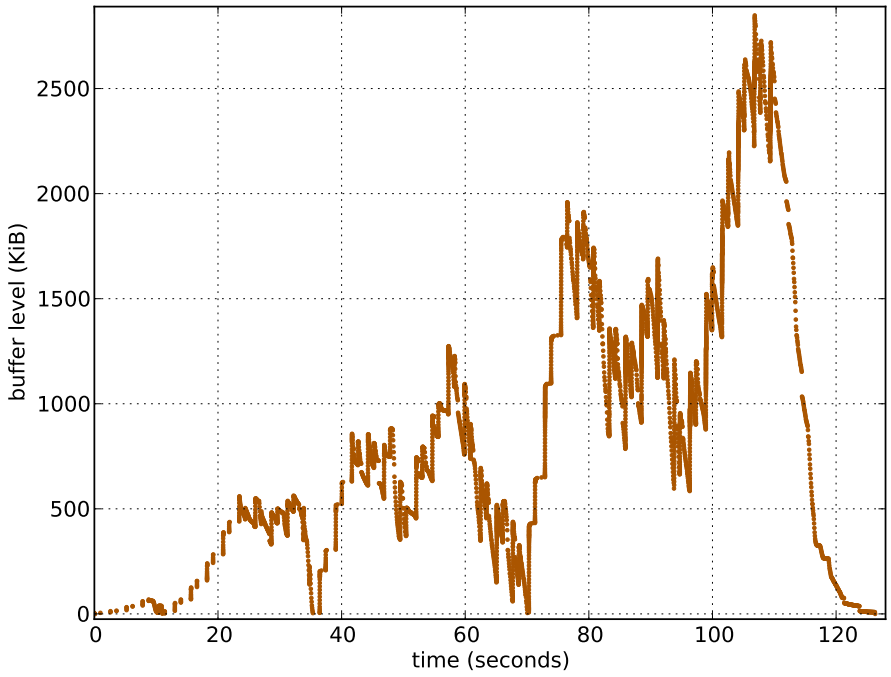


Figure 2.4: Sample buffer fill level for a 2 s and 5 s buffered video duration threshold strategy; 34 s total stalling.

2.3.2.3 *Pacing Strategies*

For segment-based HTTP streaming, a two-threshold strategy is not the only supplemental option on top of simple streaming. Here, the controller can pace the request of future segments to match the overall, or the current video bitrate. A safety margin can also be factored in to even out short time fluctuations of either the transmission or the video bitrate. For example, the controller would request segments with an overall transmission rate of 1.25 times the video bitrate. The pacing rate can either be statically chosen in advance or can be calculated dynamically based on current or future conditions. The latter leads to predictive strategies.

2.3.2.4 *Predictive Strategies*

In predictive strategies, knowledge of the future of the streaming process is used by the controller to adjust the start/stop and segment retrieval conditions. Instead of global knowledge, heuristics can instead attempt to approximate a future state.

A very simple predictive approach is to prolong the initial delay to the point, that no intermediate buffer underrun and thus no stall will occur. With global knowledge, the controller can start the stream at the earliest possible point in time, thus minimizing the total stalling time while still having only the initial delay.

Figure 2.5 depicts the time series of a sample implementation of this delayed playback predictive strategy with all necessary stalling occurring upfront.

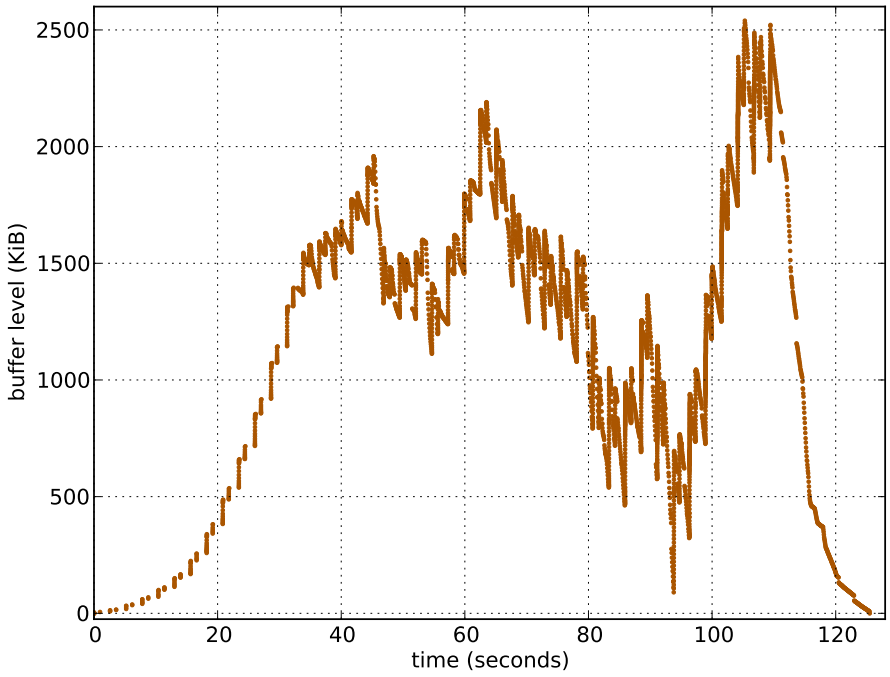


Figure 2.5: Sample Buffer fill level for the delayed playback model, 33 s total stalling.

2.3.2.5 *Adaptive Strategies*

Most strategies for adaptive streaming are an extension of both the threshold as well as the pacing strategy. However, instead of a simple transmit-or-no-transmit ruleset, they can make much more fine-grained adjustments. The quality of the stream segment to be requested will be chosen, depending on the current fill level and drain rate of the buffer. This makes a trade off between maintaining a certain quality level and putting up with increased waiting times, and dropping the quality to a level sustainable at the current transmission rate.

2.3.2.6 *Real World Implementation Examples*

Actual streaming player implementations often do not implement only one of these strategies, but rather combine ideas from several. Herein, thresholds are often set arbitrarily through the implementor's best practices and not empirically evaluated, often making a trade-off between user perceivable quality and resulting server load. In general, every streaming service practically implements its own playback strategies. This section describes three example applications.

2011 YOUTUBE FLASH PLAYER BUFFERING STRATEGY Google's video streaming site YouTube is constantly changing its appearance and technical makeup. In recent years, YouTube streams are delivered by one of three players: The Website's Flash player, a browser-integrated HTML5-based player, or custom player implementations in mobile phones, set-top boxes and similar devices. Here, the Flash-based variant used in 2011 is described.

This player used a single threshold strategy with different threshold values for the initial delay and any subsequent stalls. The values were already described in Figure 2.4. This model assumes sufficient network conditions in the beginning, requiring only a short initial playback delay to pre-fill the playback buffer. If, however, stalling occurs, then it will buffer longer to keep the stalling frequency down.

Table 2.2: Transmission Related Parameters from YouTube's Video URL Setup

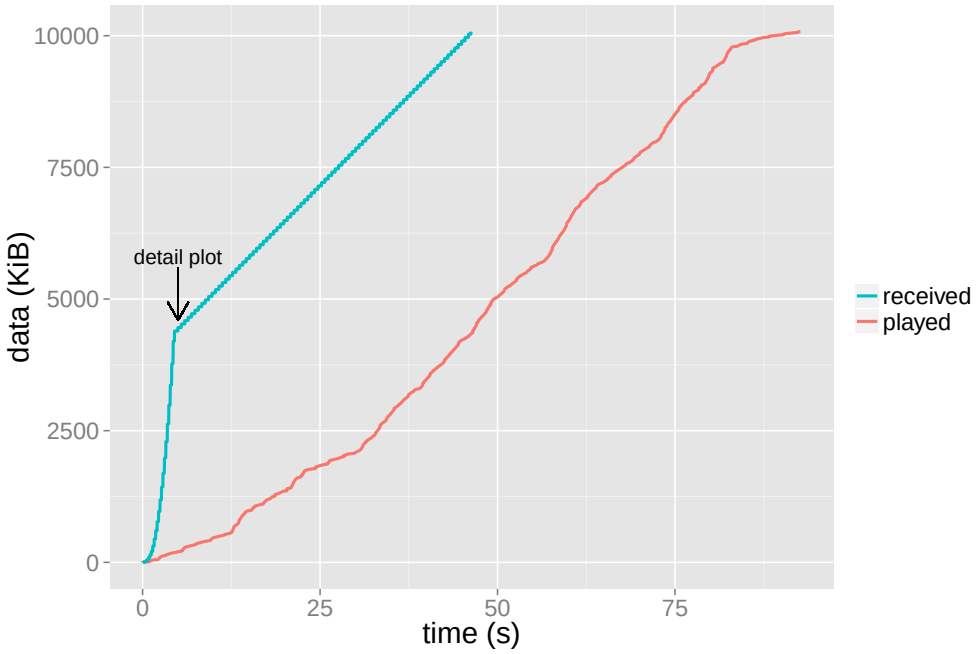
URL Part	Description
`vα.lscacheβ.c.youtube.com`	Cache server involved in the delivery.
`algorithm=throttle-factor` and `burst=40` and `factor=1.25`	Indicates initial burst plus block sending configuration.
`ratebypass=yes`	Parameter to indicate no rate limiting.

Furthermore, YouTube employs a proprietary pacing mechanism outside of the control of the streaming player. This is hinted at in the encoding of the Uniform Resource Locators (URLs) of the video files and enforced by the video file cache server. Some of those not user-changeable parameters are described in Table 2.2. The pacing was in effect for all videos below high definition resolution, but has since then been extended to include all video files.

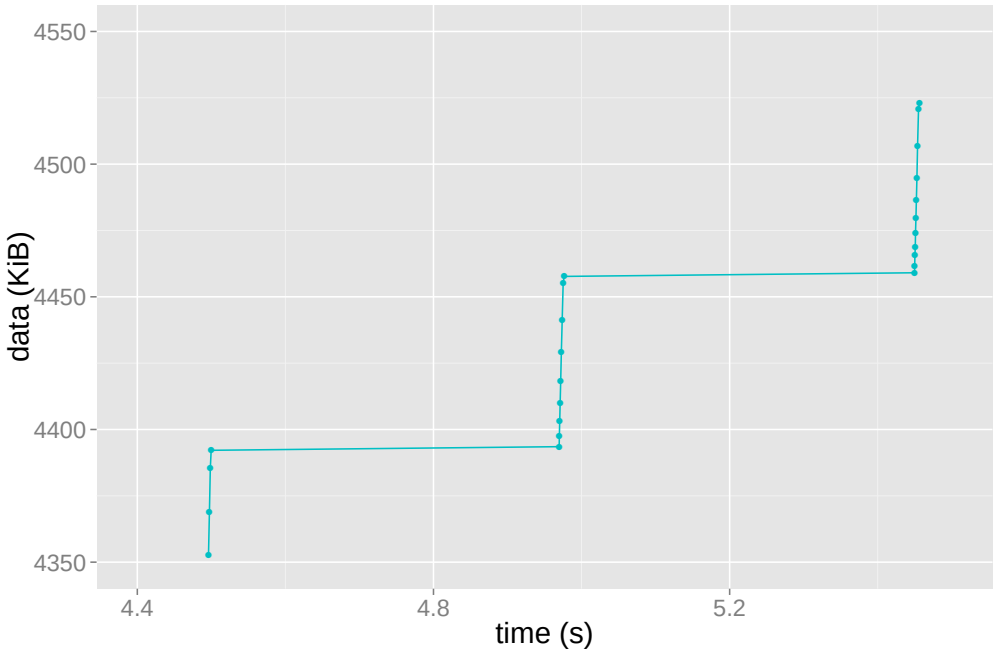
The throttling method, also observed in [AN11], limits the transmission to a rate slightly above the average media bitrate, measurements and the URL scheme indicates this to be a factor of the video bitrate of 1.25. The rate limit is not constant, instead a ON-OFF block-sending scheme is facilitated. The scheme transmits short packet bursts, typically 64 KiB in size, followed by long pauses as seen in Figure 2.6. The pause length between two bursts is set dynamically to reach the targeted bitrate on a larger time scale. The initial phase of the stream transmission is conducted unthrottled at line speed, presumably to allow for some pre-buffering to occur at the client media player. A possible reason for this server-side pacing is to avoid load spikes, with the added side effect of keeping the clients' buffer sizes in check.

FIREFOX'S HTML5 PLAYER STRATEGY Video streaming can also be directly conducted with the Web browser, through the use of a HTML5 canvas element. The World Wide Web Consortium (W3C) specifies the default technical process of HTMl5 video streaming in [Ber+13] and essentially suggests a predictive strategy. Herein, the Web browser should estimate and correlate the transmission rate to the video bitrate. A property called "autoplay" uses this definition to start playback of the associated video "as soon as it can do so without stopping". The HTML5 strategy also allows to limit the buffer size through transmission pacing, negotiated with the server, and appropriately timed range requests.

The open-source Firefox browser represents an implementation of this specification and substantiates it further. The description of this strategy is based on Firefox's version 4.0 released in March 2011. Because it is an online algorithm which does not have global knowledge of the video and transmission speeds of any point in the future it has to estimate these. To estimate the current and future rates, the moving average of the transmission rate s_{MA} and the video bitrate v_{MA} are calculated. The condition c Firefox



(a) Overall graph.



(b) Detail plot of the rate-limited block-sending phase.

Figure 2.6: Comparison of downloaded and consumed data volume revealing the pacing mechanism used by YouTube.

uses to start and resume the playback process is given in Algorithm 1, with the buffered video duration b_b, and the duration spent buffering b_T.

Algorithm 1 Firefox playback (re-)start decision algorithm.

if $s_{MA} > v_{MA}$ **then**
  $c \leftarrow (b_b = 20s \vee b_T = 20s)$
else
  $c \leftarrow (b_b = 30s \vee b_T = 30s)$
end if

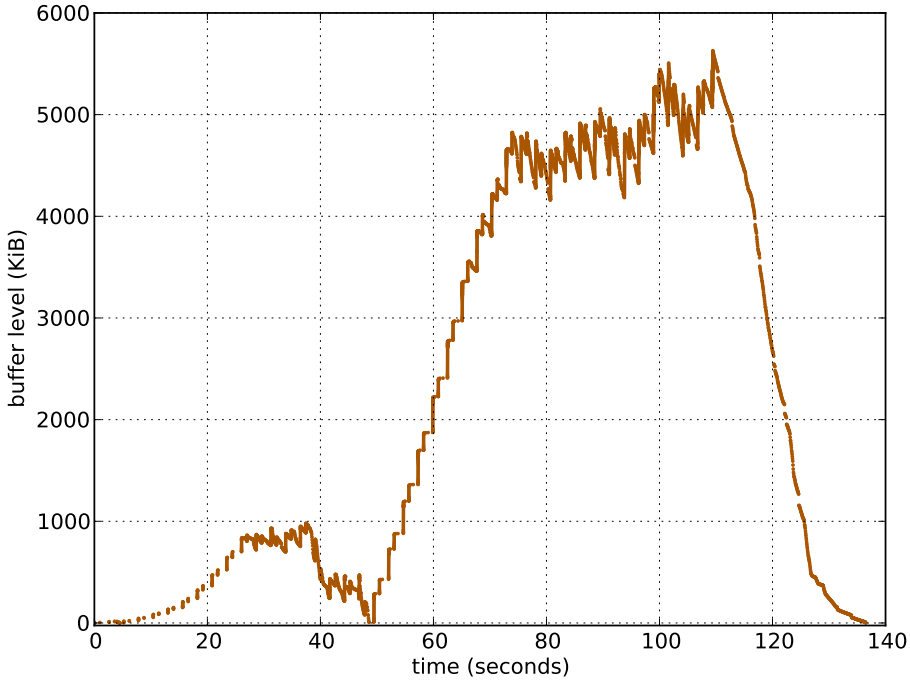


Figure 2.7: Sample buffer fill level for the Firefox 4 strategy, 44 s total stalling.

This approach is quite conservative and trades off long stalling periods for fewer stalls. The test case for our model is shown in Figure 2.7. The playback starts only after a long waiting period and intermittent stalls cause a long buffering period. Due to the longer overall stalling time the player needs to buffer more data than other strategies. This may make it unsuitable for devices with sparse amounts of memory, e.g. mobile phones. On the other hand, could a large buffer also increase the chance of continuous playback in such scenarios with bad QoS.

ADAPTIVE STREAMING STRATEGIES Implementations for adaptive streaming players are again mostly proprietary and their behavior has to be derived from measurements. This is even true for adaptive streaming protocols, which have open specifications as these generally do not specify the player's behavior.

Microsoft's Silverlight player's strategy is described in [BER11]. It employs a two-threshold model and rate estimations. When on of the thresholds is reached, the quality

will be adjusted by one step upwards or downwards as long as the transmission rate is sufficient.

The buffering behavior of further protocol variants, including Adobe's HTTP Dynamic Streaming, Apple's HTTP Live Streaming and a sample implementation of DASH, are investigated in [MLT12; ABD11].

2.4 MEASUREMENTS

With these playback models at hand, this section demonstrates how to conduct actual evaluations of reliable streaming protocols with it.

As discussed, there are numerous incarnations of reliable streaming protocols in use. Almost all of them follow the same basic approach but with slight variations in execution and choice of playback strategies and corresponding parameters. But it is exactly these choices that can have a large impact on the streaming process and resulting quality.

The problem lies in comparing these protocols to each other. Each of them is usually tied to a specific – and most often proprietary closed source – streaming player. Setting up all these players in one testbed is a huge effort and requires very specific software environments to be used on the client machines. Moreover, these players are built with user interaction and not automation and directly measuring the outcome in mind. This can still be achieved through extensive workarounds, but must be tailored to every player application. The presented approach avoids this hassle and provides a concise way to test any conceivable playback strategy in one single test setup.

2.4.1 *Progressive Streaming Measurement Framework*

To enable quick evaluations for progressive streaming the framework follows a two phase approach, separating the active online recording phase from the passive playback emulation. Recording data is very time intensive and cannot be sped up when conducting investigation of a real world process, and not relying on simulated data. It still replicates the steps a user would perform to consume a media stream on a playback device. Through appropriate configuration different scenarios can be modeled, e.g. network conditions, behavior and specifics of the user device.

Figure 2.8 depicts the framework for a streaming evaluation testbed. In phase one the stream's transmission is conducted and recorded on a per-packet level. Stream data is transmitted to the client from a server which can be any actual streaming service on the Internet or a local server under the testbed's control, eliminating undesired side effects caused by the Internet connection.

The traffic is directed through a network emulation node capable of altering the network QoS parameters, i.e. latency, jitter, and packet loss. The parameters can be set according to stochastic models derived from actual network architectures. Instead of network emulation, any preexisting architecture can also be placed here to achieve more accurate results for the intended target. This is especially helpful for complex infrastructures hard to model or with no good and concise models available yet, for example mobile and mobile core networks with the influence of encapsulating traffic into bearers.

The measurement host downloads and records the video stream as a network trace. For progressive HTTP streaming, a single request on the video file is issued and the

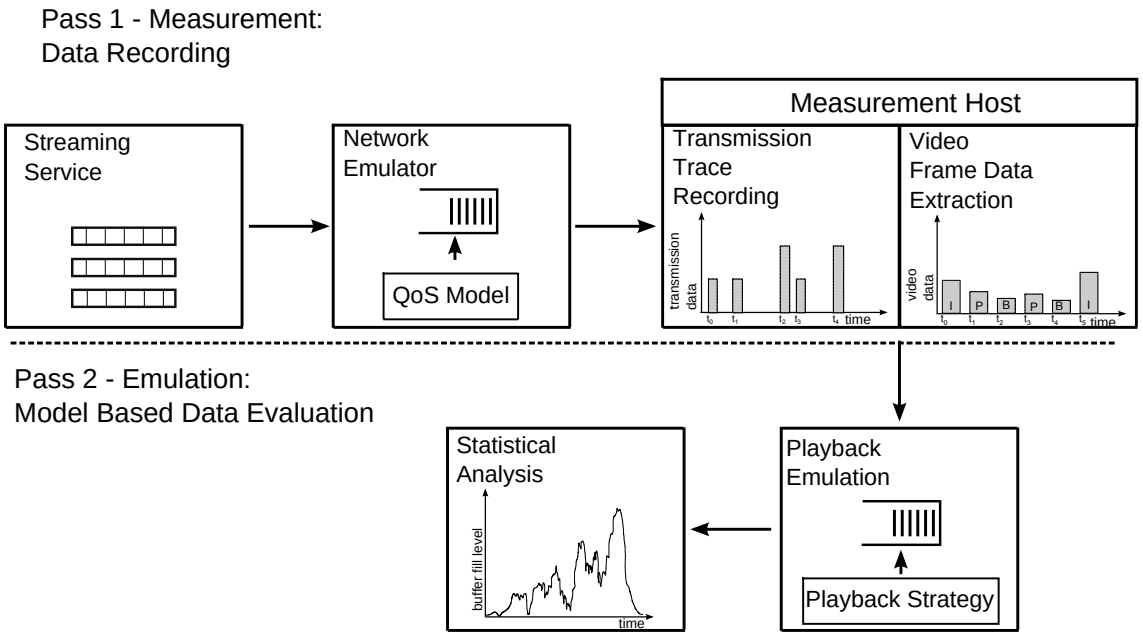


Figure 2.8: Measurement framework for progressive streaming playback strategies overview.

TCP connection maintained until the file has fully arrived. This process is recorded as basis for the second phase. These traces should consist of the size and timestamp of every incoming packet. More detailed traces can be used to scrutinize other layers of the connection, e.g. the dynamics of TCP receive window size. Additionally the received file is decoded yielding a trace of all video frame sizes and playout timestamps.

In the second pass, these two traces are then used to feed the actual reliable streaming playback model described before. This conducted by a closed-loop emulation process calculating the current buffer fill level based on the collected transmission and video frame traces for every point in time.

All of the progressive streaming – i.e. non-adaptive – strategies can be tested on the same trace set. In simple HTTP streaming, the transmission is not controlled by the streaming application and no rate control is conducted. Therefore, recording the packet trace and simulating playback are completely decoupled, as the latter cannot influence the former. This enables fast and efficient comparison of non-feedback protocols subject to the same network conditions.

The emulator then generates playback stalling statistics, specifically their number and duration, to compare the effect of the different strategies on the same trace. With these results, parameter settings for playback strategies can also be iteratively tested and improved leading to an empirical calibration of playback strategies instead of relying on best practices.

One of the drawbacks of this model-based emulation approach is of course the reliance on suitable models and playback strategies for the stream protocols under scrutiny. Obtaining these from proprietary streaming clients, can be a difficult reverse-engineering process.

2.4.2 *Adaptive Streaming Measurement Framework*

Up to this point, the measurement framework is only suitable for progressive streaming omitting any adaptive strategy. This second framework modifies this and additionally allows for the testing of adaptive playback strategies. However, to achieve this, the advantageous two phase setup cannot be employed anymore.

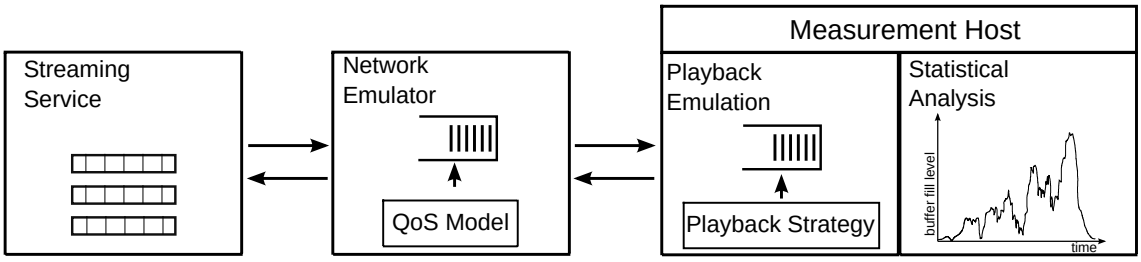


Figure 2.9: Measurement framework for adaptive streaming playback strategies overview.

Figure 2.9 shows the adapted framework. The playback emulation process is now directly fed with the stream transmission without recording it first. The emulation is now an online process and has to be conducted in realtime. This enables the emulator to react on the current streaming state and request an alteration from the server. The adaptation spectrum ranges from the timing of stream segment retrieval to the chosen quality level of future segments.

While allow for a wider range of playback strategies, this approach is also inherently slower as it does not allow a speed-up beyond realtime, limiting its usability somewhat. Therefore, a transition to a full simulative approach is suggested. This is path is further explored and discussed in Section 4.3.3.

[TODO: check simulation approach ref or further explain simulation]

2.4.3 *Technical Implementation*

To conduct measurements, the described two phase progressive streaming measurement framework was implemented in a testbed. This testbed consists of three interconnected physical nodes running Linux.

The optional streaming server houses an Apache httpd[6] Web server, hosting the files that are to be streamed. Alternatively, Internet streaming service traffic can also be directly routed through the network emulation node.

The network emulation node uses the existing QoS capabilities of the Linux kernel, dubbed NetEm [Hem05], to add latency and packet loss to the transmission as well as to act as a bandwidth bottleneck. The additional delay is set deterministically, the loss follows a uniform distribution without any correlation in the transmission.

To both retrieve the streaming file and record the transmission process at the client, curl[7] is used. If so desired, tcpdump[8] can also be facilitated to achieve a higher recording precision. The video file is then parsed for its frames and sizes using mplayer[9] with

6 https://httpd.apache.org/
7 http://curl.haxx.se/
8 http://www.tcpdump.org/
9 http://www.mplayerhq.hu/

libav[10]. The traces are then put into the actual playback emulation, implemented by custom Python-based code and statistically evaluated with Python as well as R.

2.4.4 *Measurement Series and Evaluations with the Framework*

This implementation has been used to conduct a comparative study of two theoretical and two real world playback strategies. They are tested for their susceptibility to worsening network QoS, latency and loss in this case. The evaluated strategies are the described YouTube and Firefox strategies as well as the null strategy and the predictive strategy with perfect knowledge and just the initial stalling period.

The video used in the experiment was streamed from the YouTube web site providing a realistic foundation for the experiments. This also enables a server side pacing mechanism adjusted to the video bitrate for free. Details on the video used in the experiment are available in Table 2.3. Two measurement series are performed with this video, both only differ in the network emulator setting. The first series increasingly adds packet loss to the stream, with the second series altering the packet delay. In both scenarios, the link bandwidth was limited to a typical dialup value of 16 Mbit s^{-1} in the downlink direction and 1 Mbit s^{-1} up.

It can be stated, that all playback strategies will generally work very similar under good network conditions as long as the TCP goodput, i.e. the rate the payload is transported by TCP, is higher than the video bitrate. With sufficient goodput video streams will start with almost no delay or intermediate buffering. If, however, the achievable TCP goodput is close to the average video bitrate, the buffer can be quickly drained by short deviations from the average transmission and video bit rates. The TCP goodput can be limited by high latency and loss. Many TCP congestion control algorithms depend on the round trip time. If the Round-Trip Time (RTT) is high, the congestion window will increase much slower. High latency triggers TCP timeouts and retransmissions, and in turn decrease the congestion window reducing the goodput. Packet loss affects TCP goodput even more. A lost packet results in duplicate acknowledgments followed by retransmissions and a decrease of the congestion window. The problem is worsened if the acknowledgments are also lost. The connection could stalls on missing old segments without which the playback cannot proceed. In addition to the reduction of goodput this results in a delay burst and high jitter for the streaming application.

Table 2.3: Test Video Parameters

Parameter	Value
Video Duration	92.536 s
Size	9.61 MiB
Framerate	23.976 s^{-1}
Average Video Bitrate	871 kbit s^{-1}
Codecs	AVC+AAC

10 https://libav.org/

In the latency measurement series, the emulator delays forwarding the packets for a constant amount of time. The latency was increased in 100 ms steps, up to a total of 5000 ms. The added latency is split up evenly between the uplink and the downlink.

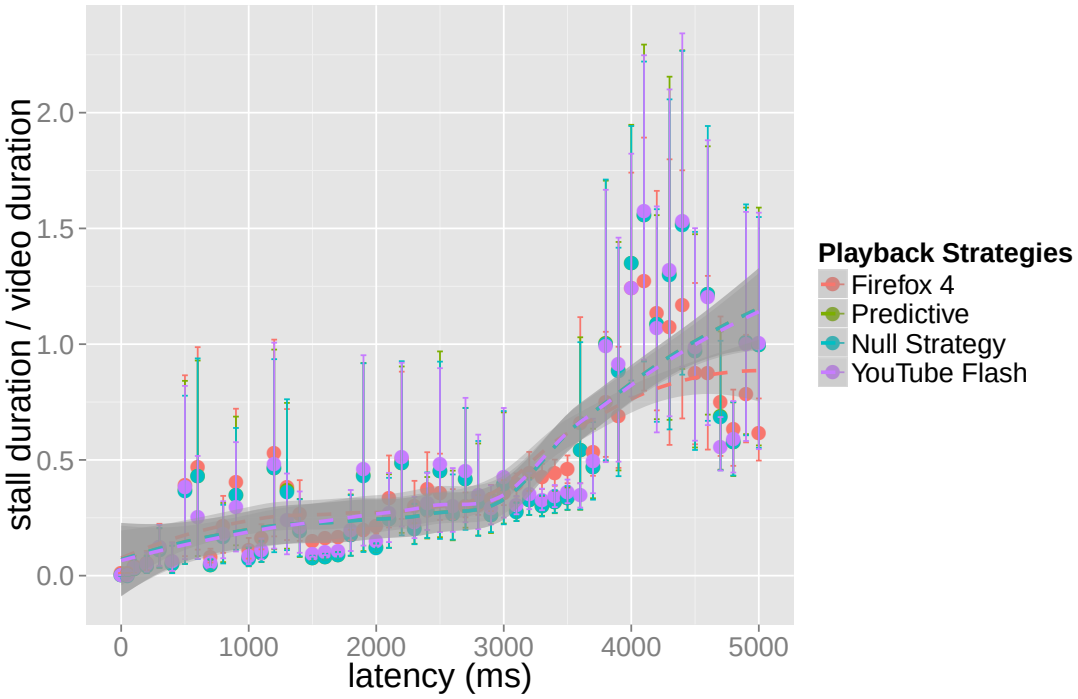


Figure 2.10: Stalling duration in relation to transmission latency with local polynomial least-squares fit.

Figure 2.10 depicts the relation between the added latency and the stalling duration of the playback strategies. The stalling time increases as expected with the additional latency, but Firefox's strategy seems to have slight edge during high latency. Overall, the stalling duration quickly reaches a length comparable to the actual video duration and even surpassing that. Someone watching a stream under this condition might find this not acceptable any longer.

Figure 2.11 additionally shows number of stalling events occurring during the playback with the null strategy at the high end and the predictive strategy just showing the expected single stall before playback start.

Overall, it can be said, that in this specific latency scenario the real world playback strategies seem to honor the fact that more video interruption lead to a worse experience than fewer but longer ones. Through mobility and handovers, mobile devices experience short bursts of latency of several seconds fairly common. According to the measurement series, the resulting stalling behavior could still very well be bearable for streaming users.

In the loss measurement series, uncorrelated and uniformly distributed loss was added in both the uplink and the downlink direction. The loss was incrementally increased in 2% steps up to a total additional loss of 12.5%. About 4% of the individual experiments did not finish correctly, and the video was not completely transmitted. This was caused by the TCP stack, which at some point terminated the connection after to many packets were lost back to back, and curl giving up after several retries. This unpre-

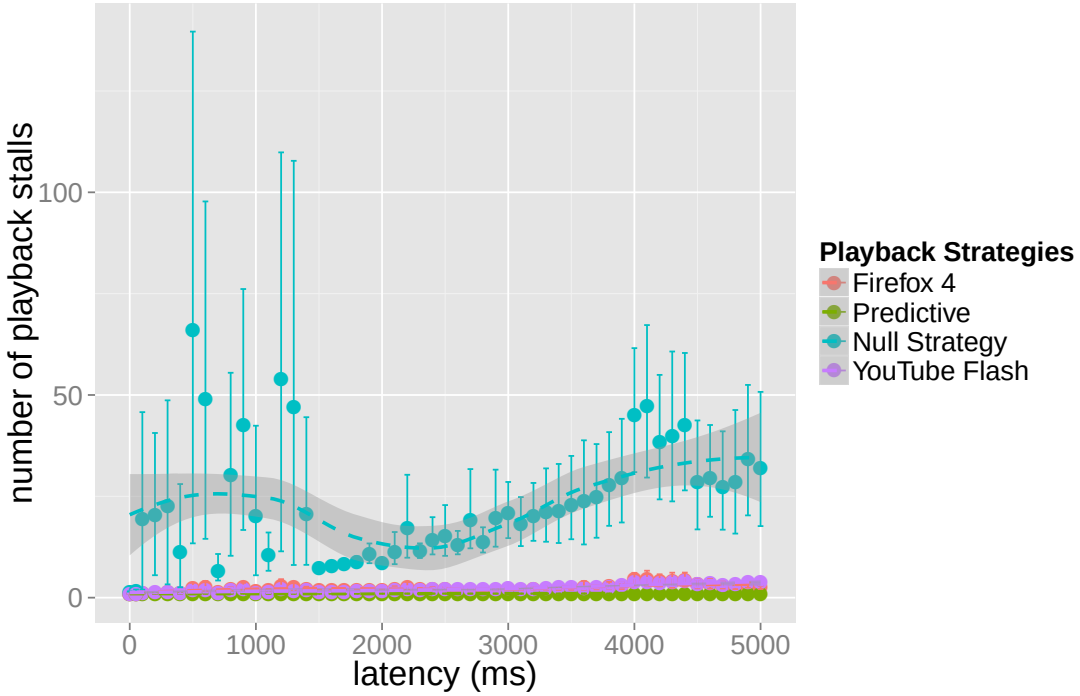


Figure 2.11: Number of stalls in relation to transmission latency with local polynomial least-squares fit.

dictability and failure rate also leads to the high variability seen in the results of the loss measurement series. Nonetheless, a certain trend can still be derived from the results.

Figure 2.12 shows the resulting relative stalling duration in the packet loss measurement series. Loss of up to 4% seems to have no discernible impact on the streaming process. Anything beyond this point sees a large increase in the stalling duration. With a relative stalling duration of almost four times the actual video length at 12% packet loss any streaming attempt is practically rendered unusable. This behavior could hint to the transport protocol's reliable transport feature, catching any occurring loss. However, the detection and retransmission takes time and leads to a bursty increase in latency offering a possible reason of the increased stalling time.

Figure 2.13 shows the extremity of the null strategy in terms of the number of experienced stalls compared to other strategies reaching a number two orders of magnitude larger than any other model.

As a result, when planning a network for streaming applications, the maximum loss should be kept below the 4% mark to achieve reasonable streaming quality.

[TODO: expand loss section a bit more]

2.5 STREAMING SUMMARY

In the course of this chapter, a complete toolset to classify, model, emulate, measure, and finally to evaluate video streaming and interpret the results was given.

The multitude of approaches to reliable HTTP streaming in recent years, required these tools. Previous advances proofed to be not flexible or fast enough to deal with this influx of new system parameters. The presented measurement framework based on a

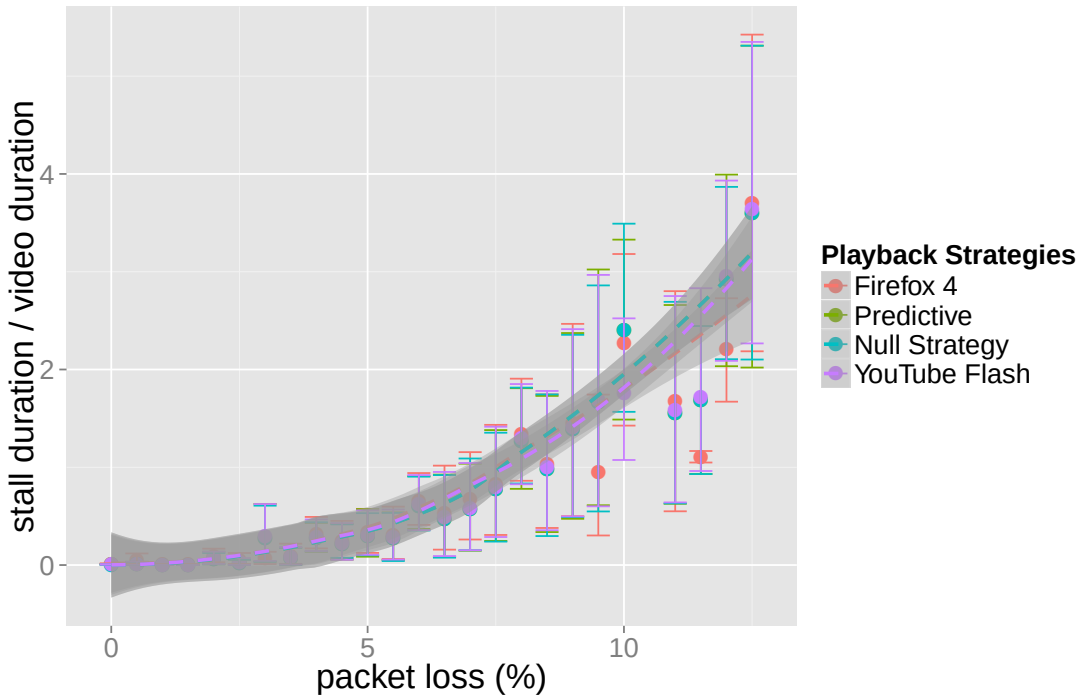


Figure 2.12: Stalling duration in relation to packet loss with local polynomial least-squares fit.

simple playback model can be an answer to this issue. Implemented as a emulation in a testbed or a DES it can deal with almost any reliable streaming protocol.

Out of these reliable protocols, a differentiation and categorization solely based on their playback strategies is made and compared to real world examples. With the help of the measurement framework and testbed, these strategies were quickly evaluated for their behavior under difficult network QoS conditions. Although they might not behave perfectly when stressed, they probably will still be the way forward in the future, because of their reduced complexity and the shift of control logic from the server to the client as well as from the protocol to the application.

This also makes them very interesting for usage on mobile device, the young application ecosystems growing there, and the very distinct behavior of mobile networks in contrast to classical wireline networks. This will be part of the next chapter's investigation.

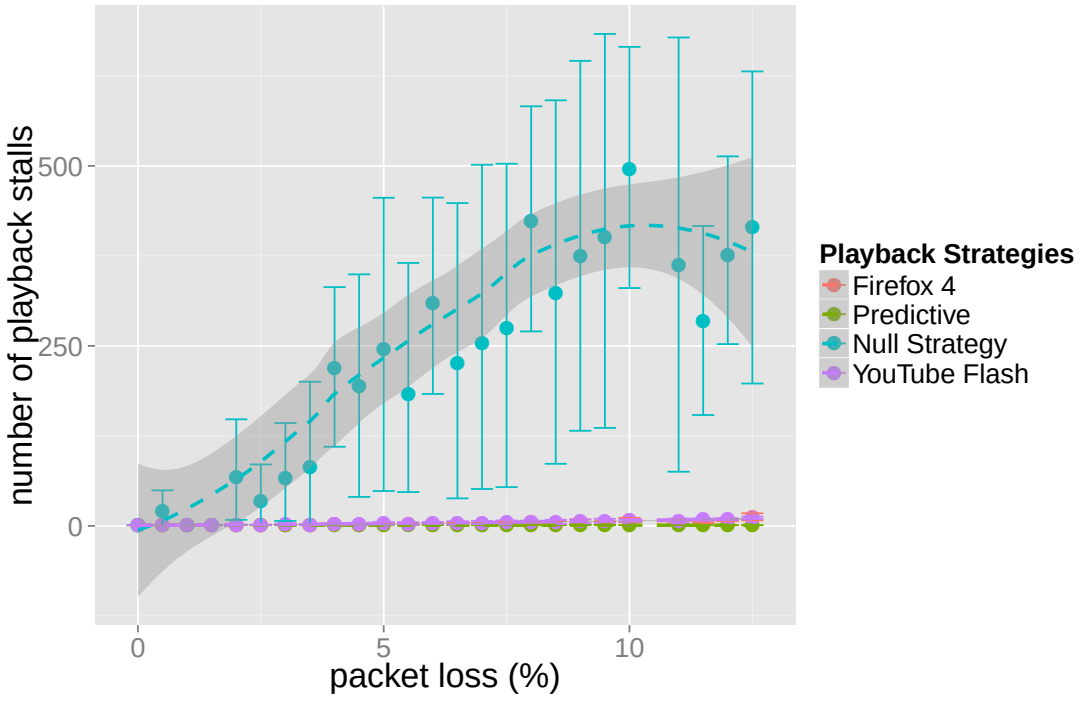


Figure 2.13: Number of playback stalls in relation to packet loss with local polynomial least-squares fit.

3

MOBILE CORE NETWORK INVESTIGATION

The rise of reliable video streaming is not happening completely independent. Instead, it is deeply embedded into major shifts in Internet access. With more and more smartphones being used and even dial-up Internet access sometimes being replaced with stationary mobile 3G and LTE modems. All these "over-the-top" streaming services are now being used within mobile networks.

These cellular mobile networks are usually based on 3GPP specifications which have evolved from the circuit switched GSM network into the fully packet switched LTE which is still in its unrolling phase. But being packet switched does not mean that it shares a lot of similarity with a typical wireline Internet protocol stack and network infrastructure, for example a Very-high-bit-rate Digital Subscriber Line (VDSL) or Data Over Cable Service Interface Specification (DOCSIS) dial-up connected to an IP network. A "3G" network (a term synonymous for the typical type of cellular network used today) is very distinct from typical wired networks as it provides, amongst others, mobility and authentication in its lower layers through the core specifications rather than being optional on-top services as is typically the case in the Internet.

The TCP/IP stacks largely follows two principles: "Keep it simple, stupid" (KISS) and the end-to-end principle[SRC84], which essentially means to restrict the protocols to the necessary bare-minimum and keep state only in the end systems. 3G takes a different approach, keeps a large amount of state at the obligatory nodes in its "core network", which explicitly communicate by signaling procedures defined by the 3GPP. The adverse effects of state-keeping in network devices have been known to, e.g., Internet users running BitTorrent across low-end home routers as of the early 2000s. In UMTS mobile networks, the networking hardware is vastly more powerful, but the control plane tasks are vastly more complex than port and network translation as well, namely carrying and routing IP and voice traffic, user mobility, Authentication, Authorization and Accounting (AAA) and so on. Many specialized protocols are involved to communicate intents and states in the network. This causes processing overhead, additional traffic on network paths, and increases the number of states to be held in memory on the core network nodes. All of these attributes can be subsumed under the term network "load" which we plan to investigate in this work.

Given its roots as a research network and its growing pervasiveness over the last forty years, it is understandable that research on the Internet covers all parts of the network from applications to access to the core, and has been going on ever since what could be considered prehistoric times of the Net. The state of research on mobile cellular networks such as 3G is lean in comparison. Mobile networks providing Internet access have not been around for too long, and still are not available in all parts of the world. Furthermore, most research focuses on user-oriented metrics such as traffic statistics and mobility patterns, or takes into account the radio part of the network only. Little has been published about activity within the core network, and yet less about signaling.

Given how limited spectral resources on the radio interface are, it might not seem obvious to think about signaling load in the network. Yet, there have been situations where the core network unintentionally has been flooded with signaling, taking down

user-plane connectivity on the way, despite small amounts of actual user traffic being transported [Don12; Cor11].

The adverse effects of state-keeping in network devices have been known to, e.g., Internet users running BitTorrent across low-end home routers as of the early 2000s. In UMTS mobile networks, the networking hardware is vastly more powerful, but the control plane tasks are vastly more complex than port and network translation as well, namely carrying and routing IP and voice traffic, user mobility, AAA and so on. Many specialized protocols are involved to communicate intents and states in the network. This causes processing overhead, additional traffic on network paths, and increases the number of states to be held in memory on the core network nodes. Therefore, in scenarios such as the ones mentioned above, radio access is not the bottleneck to connectivity any more, but signaling is.

This chapter is structured as follows. As it is absolutely necessary to get a grasp of the finer details of the mobile network architecture under investigation

- Technical Background. Standards and Specifications. Network Architecture. Protocol Details.
- Dataset Description. Methodology.
- Definition of Load etc
- Evaluations. Dataset Analysis. Statistics. Traffic Characteristics. How does that user traffic fit in?
- Fitting, Modeling and Simulation
- Summary

Core is usually neglected in communication investigations

While other publications look at the near-edge interactions in these network, research on the core is scarce, the reason for it being simple: you cannot do research without data from the operator there. Research at the edge, beginning at the IP stack level and upwards, can be conducted relatively simple. Writing simple tests and measurement scripts, often involving tcpdump and other tools, is usually all you need. But a mobile phone doesn't let you peek inside its layer 1 and 2 interactions (or even the implementation). Any information on this black box must be indirectly inferred from above (forcing behavior known from the specifications through scripts) or below (spectrum analysis using software defined radio approaches). To take a look at the core's view of traffic and data, one needs access to a dedicated measurement and capturing infrastructure placed inside the network. With this, researchers can not just look into user traffic flowing through the network but also quite easily into the signaling heavy mobile network control plane.

Operators usually dimension their networks in relation to the occurring user traffic. But in such a signaling-dependent architecture this might not hold true anymore, as every user traffic has to be explicitly allowed, set up, and metered through all of the network's components. This has already led to some troubles in some mobile access networks, as heavy signaling for user traffic tunnels with very small amounts of traffic, that were however closed and reopened at a very high rate, caused an unintended Distributed Denial of Service (DDoS) in the radio access network[Don12; Cor11]. This

inherent complexity of signaling in mobile cellular networks is easily missed by programmers who do not or cannot know that their applications will run over such wireless links, and probably would not expect it from a network that pretends to transparently carry IP.

In this publication we attempt to give some insights into the mobile network control plane and its impact on dimensioning and load modeling. To do this, some important aspects of the 3GPP specifications have to be explained to give some basic vocabulary for the following exploratory research into signaling with a focus on Packet Data Protocol (PDP) Contexts and their management through GTP tunnel management procedures. Using a week long data set from an Austrian mobile operator recorded at the Gn interface between the SGSN and GGSN, we attempt to find criteria influencing the signaling but we are also formulating hypotheses on the load impact of signaling, both backed by statistics gathered from the the data set.

The inherent complexity of signaling in mobile cellular networks is easily missed by programmers who do not or cannot know that their applications will run over such wireless links, and probably would not expect it from a network that pretends to transparently carry IP. What furthers this problem is the lack of literature on the theoretical and practical sides of these issues.

This apparent lack is due to a number of reasons. First, gaining sufficiently intimate knowledge on the huge corpus of 3GPP Technical Specifications is a laborious task. Second, to come up with lower-layer measurements requires physical access to the core network infrastructure and suitable measurement equipment. Also, much of the data is commercially and privacy-sensitive, and cannot be published without extensive sanitizing.

The purpose of this paper will therefore be to give a 3G tunnel management primer, introducing the relevant GPRS/UMTS network structure and the involved control plane protocols with a special focus on the GTP, which is probably the most prevalent. Furthermore, we share our first insights into one practical aspect of the signaling process, the GTP tunnel management procedures. Using a week long data set from an Austrian mobile operator recorded at the Gn interface between the SGSN and GGSN, we take a look at PDP Context durations, i.e. the time a PDP Context is established and held, argue how this influences the load on the network, and evaluate the data by device types and operating systems.

Our measurement data backs up a number of straightforward assumptions on the behavior of different device and operating system types, but also reveals some remarkable differences in tunnel characteristics.

3.1 MOBILE NETWORK ARCHITECTURE

This section starts with a primer on cellular data network basics, and then moves on to describe relevant details of GTP, the tunneling protocol under investigation.

Today's dominating commercial mobile network system, which includes GSM, UMTS, and up to LTE, is designed and specified by the 3GPP. This group is an umbrella organization for several standardization bodies, including the European European Telecommunications Standards Institute (ETSI) and their individual members — in this case mostly

telecommunication companies. Unlike the IETF, the Internet's de facto standards body, natural persons cannot participate in the 3GPP on their own but only through the organizational members.

Specifications are not released individually but are instead grouped together into larger releases once every or every other year. GSM was first specified in the Phase 1 release in 1992 with GPRS added in Release 97 (1998). UMTS followed with Release 99 (2000), but most 3G networks operate at least with Release 5 (2002), 6 (2004), or 7 (2007) as they introduced High-Speed Downlink Packet Access (HSDPA), High Speed Uplink Packet Access (HSUPA), and HSPA+ likewise. LTE first found its way into the specifications in 2008 with Release 8 with Release 12 currently being worked on. This background section mostly describes the UMTS based Technical Specification (TS) with some comparisons being made to the older GPRS and the latest LTE specification versions where available.

Today's most common version of the mobile network architecture is depicted in Figure 3.1 and based on 3GPP TS 23.002 [3GP13g], with some minor nodes and network paths omitted[1]. The displayed architecture combines all three access technologies as well as the Circuit Switching (CS) and the Packet Switching (PS) domain of the core. Concerning the PS domain, one has to further distinguish between links and nodes used solely for control plane tasks as well as links and nodes that are in path of the actual user IP traffic. For UMTS and GPRS[2] the SGSN and the GGSN are the core in-band elements.

One thing to keep in mind is the strict separation between user plane and control plane tasks in the 3GPP architecture. Usually completely separate protocol stacks are used and signaling is mostly very explicit and out of band. This is contrary to the typical approach of the Internet's stack, especially its upper layers, where most state is implicitly inferred and only some signaled in-band. The following paragraphs give a short description of nodes, general and PS tasks handled by them, and used protocols and stacks in the two major network domains inside the infrastructure domain [3GP12b] for both the user and the control plane. The description will be mostly focused on the UMTS parts of the architecture

3.1.1 *3GPP Radio Network*

The architecture has three completely distinct radio networks, one for each access technology: GSM's Base Station Subsystem (BSS) (or more complete: GSM/EDGE Radio Access Network (GERAN)), UMTS Terrestrial Radio Access Network (UTRAN) for UMTS, and Evolved UTRAN (E-UTRAN) in LTE.

Essential to the radio network is a base station, a radio transceiver providing the physical connection to the user's mobile device[3] 3G's base station is called Node B. The used radio spectrum is divided into a number of channels, with various shared channels responsible for management and control plane signaling and one or more dedicated channel for each active mobile device [3GP12d; 3GP12e]. Layer 2 consists of several pro-

1 For a complete reference of all the acronyms and addressing schemes used in 3GPP specs please refer to TS 21.905 [3GP13m]. and TS 23.003 [3GP13h]

2 GPRS provides PS data services for GSM radio access. The same core infrastructure is also used in the UMTS PS domain.

3 Mobile devices are called Mobile Station (MS) in GSM networks and User Equipment (UE) in 3G and later.

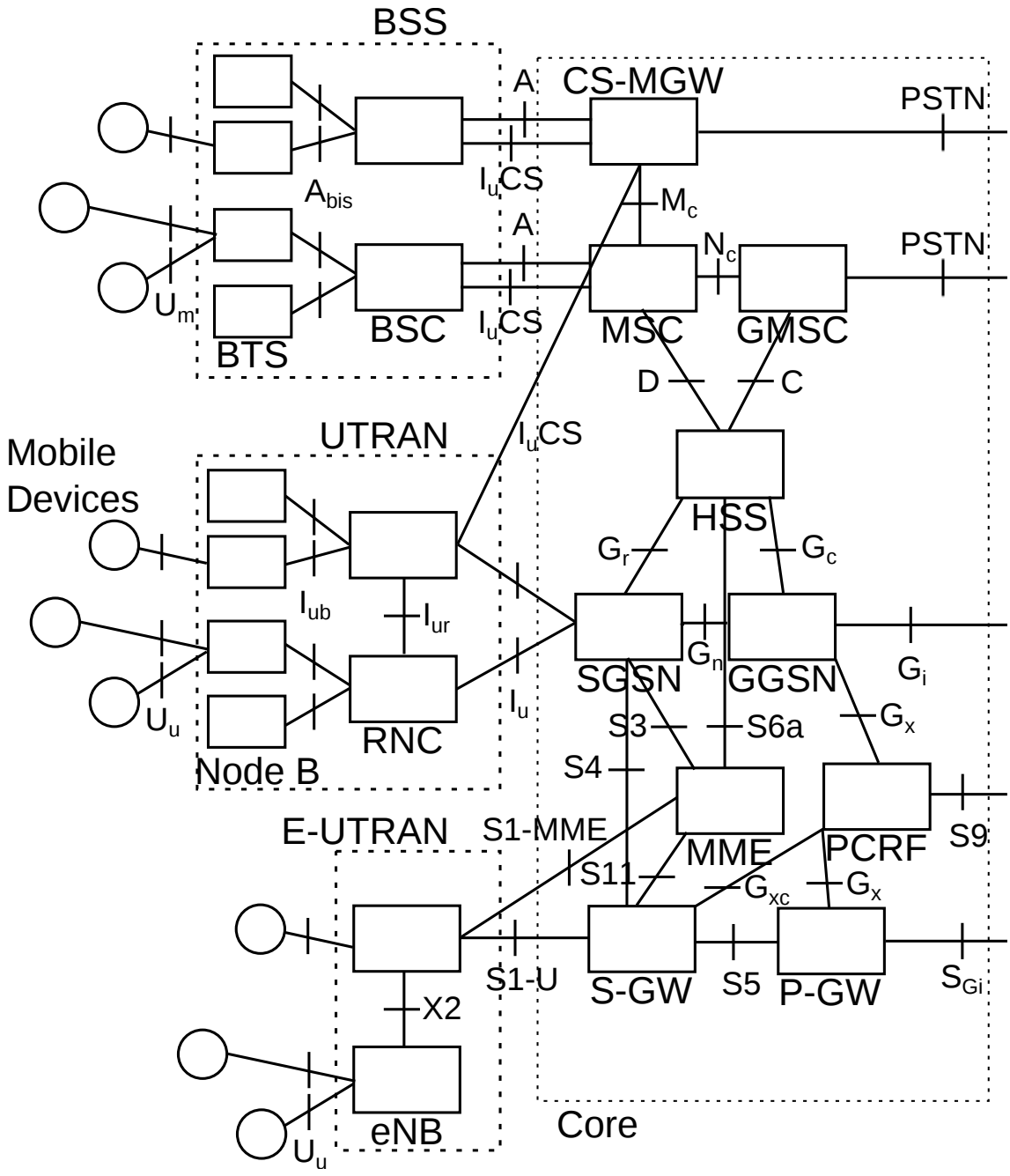


Figure 3.1: Combined CS/PS GSM/UMTS/LTE architecture overview.

tocols managing and multiplexing transmissions on the link. These are Media Access Control (MAC) [3GP08d] and Radio Link Control (RLC) [3GP08g] with an additional user plane broadcast service provided by Broadcast/Multicast Control (BMC) [3GP08a]. On Layer 3, the actual radio control plane signaling protocol RRC [3GP12f] resides, managing the device's state and the radio connection. Some of the signaling procedures are detailed in [3GP12g]. Additionally, Packet Data Convergence Protocol (PDCP) [3GP12c] provides the connection to the usual Internet TCP/IP userplane stack atop. Thus, all user traffic is encapsulated into so called *radio bearers*, which tunnels traffic from the mobile device directly into the core network.

Each base station acts as an independent radio cell. Mobile devices can be handed over between cells without higher layers being able to notice this.[4] The handover process is fully controlled and conducted by the network by a node that shares the old and new path to the device. For UMTS, in most cases this can be handled by the Radio Network Controller (RNC) while the core SGSN is responsible for handovers between regions.

In UMTS multiple Node B are concentrated into one RNC. Most of the functions of a RNC, including mobility management, are defined by the Radio Access Network Application Part (RANAP) control plane protocol [3GP08h]. RANAP is used in the Iu interface between the RNC and the core network, i.e. the SGSN. All non-radio links of the network are today IP-based, but in the past all interfaces have also been explicitly defined for Asynchronous Transfer Mode (ATM) with some differences to the IP-based stack. The connection of the decentralized parts of the radio network to either the core network itself or a RNC is called *backhaul*. This term usually subsumes the bulk transport of data over dedicated links to a central location. Often, optical fiber or microwave transmission links are used.

3.1.2 *3GPP Core Network*

The PS domain of a mobile core network manages most of the aspects of the connected devices and acts as the bridge to the Internet. It consists of nodes that are directly in the path of the user plane traffic as well as additional nodes, that only exist in the control plane.

GPRS and UMTS use the exact same PS core network architecture, only LTE introduced a new core network concept, the Evolved Packet Core (EPC). If the same core is used for both (or all three) access technologies, nodes for all architecture versions have to be present with the exception of some control plane nodes, that EPC replaces with its counterparts that provide legacy interfaces.

The two central elements in the userplane's path are the SGSN and the GGSN [3GP12a; 3GP13c]. The SGSN is the endpoint of the Radio Access Bearer (RAB), tunneling user traffic from the UE to the core, and an endpoint for the GTP based core tunnel, further transporting userplane traffic to the GGSN. GTP will be described in detail in Section 3.1.3.3. The GGSN's control plane tasks include mobility and connection management via RANAP to the glsRNC. The necessary information is cached and retrieved from the Home Location Register (HLR) using Mobile Application Part (MAP) [3GP13e]. The HLR or, respectively, the Home Subscriber Server (HSS) in an EPC, acts as the central storage of all the operator's subscriber information.

4 However, there are still many ways to detect a handover in the upper layers of a mobile device.

The GGSN represents the gateway to the public Internet and therefore typically is the only node in the network, that has a publicly routable IP address. It filters incoming traffic into the corresponding GTP tunnel and routes packets to the correct SGSN. State about any active tunnel and device has to be kept locally and is initially retrieved from the SGSN.

In EPC user traffic in the core is handled by the Serving Gateway (SGW) and Packet Gateway (PGW), having similar functions as their GPRS counterparts SGSN and GGSN. Depending on the specific implemented infrastructure these two nodes can also be combined into one, eliminating the S5 interface and the GTP signaling between them. Much of the control plane tasks of the SGSN have now been offloaded to a new node, the Mobility Management Entity (MME), which maintains its own logical connection to the radio network using the S1 Application Protocol (S1AP) signaling protocol [3GP08b]. Instead of MAP to retrieve user data from the central storage, Diameter [Faj+12] is now used to communicate with the HSS. Of note is also a further addition to the Evolved Packet System (EPS): the Policy and Charging Rules Function (PCRF) in conjunction with the Policy and Charging Enforcement Function (PCEF) integrated into the PGW. They act as a Deep Packet Inspection (DPI) entity, inspecting all userplane traffic and enable arbitrary filtering of traffic and traffic-based billing [3GP13i].

3.1.3 *Core Network Concepts and Protocol Details*

All core network signaling procedures and state machines are specified in TS 24.008 [3GP13f]

simple example procedure: PDP context activation procedure in UMTS invokes additional Customised Applications for Mobile Networks Enhanced Logic (CAMEL) procedures defined in [3GP07]

29.060 [3GP13b] GTPv1 description; GPRS/UMTS version; Gn (SGSN - GGSN) Gp interfaces 29.281 [3GP13d] GTPv1-U user traffic tunnel description 29.274 [3GP13a] GTPv2-C protocols and procedures (LTE only) on S5 (S-GW - P-GW) and S8 (P-GW - P-GW) interfaces

3.1.3.1 *Mobile Network Control Plane*

separation between control and user plane: tunneling concept state machines protocols procedures -> location of control inside the network vs at the edge comparison of explicit (3G) control networks vs implicit (IP) resulting complexity ...

3.1.3.2 *State Keeping: PDP Context Concept*

explain data structure/state

3.1.3.3 *GTP and GTP-based Core Network Signaling*

TUNNELING CONCEPT AND BEARERS LTE: mention also Proxy Mobile IPv6 (PMIP) as GTP alternative, and the option to have a combined SGW PGW

One UE Can have a maximum of 3 Public Data Network (PDN) connections and 11 bearers (one default per Access Point Name (APN) and additional dedicated).

- Every bearer has a predefined QoS level between UE and P-GW. ==> Level of Granularity for QoS control.

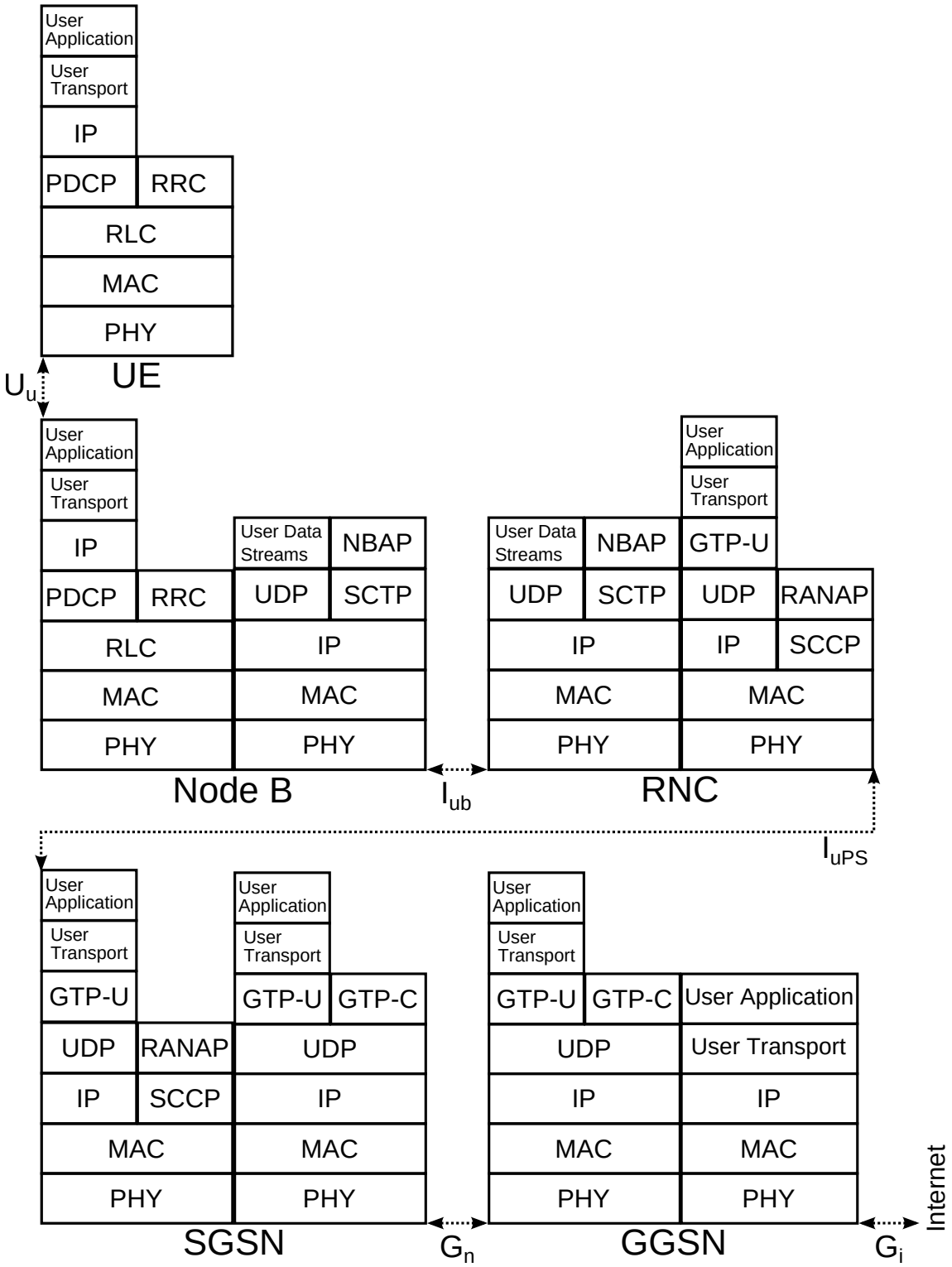


Figure 3.2: Simplified control plane and user plane IP-based protocol stacks on the user traffic path through the mobile network.

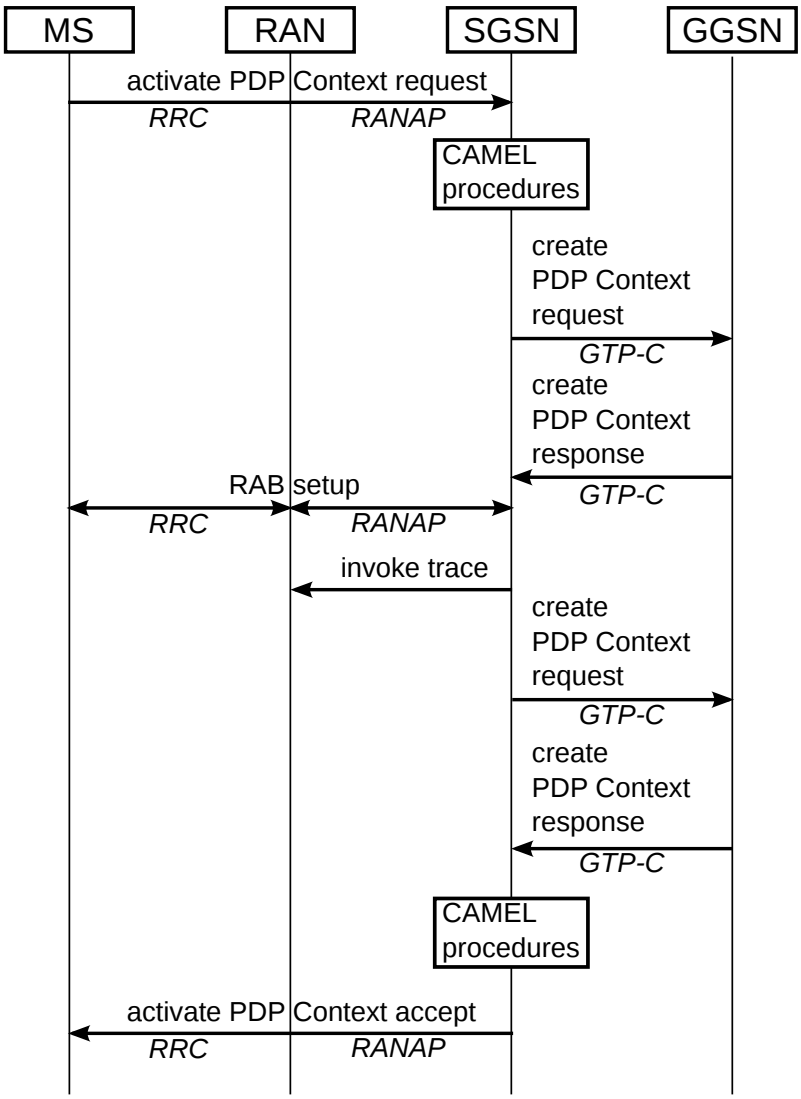


Figure 3.3: PDP Context Activation procedure signaling interaction diagram for UMTS, including involved signaling protocols.

- Initial bearer QoS level assigned by network based on subscription data.
- Guaranteed Bit Rate (GBR) bearers: dedicated network resources permanently allocated at est/mod. Otherwise Non-GBR.
- The Traffic Flow Template (TFT) belonging to a bearer is a set of packet filters that assign traffic flows to the bearer.
- UL-TFT at UE, DL-TFT at PCEF (P-GW).
- default bearer: always-on IP connectivity for the UE to a PDN
- dedicated bearer:
  - any additional bearer for the same PDN
  - Traffic Flow Template (TFT) associated with every ded. bearer
  - establishment/modification decision only by EPC

  - QoS level assignment only by EPC

- default bearer may be used as m,catch-all traffic bearer for everything that does not match any filter
- Every bearer associated with QCI and ARP.

  QoS class identifier (QCI): standardized scalar as reference for node-specific QoS parameters Allocation and Retention Policy (ARP): priority level preemption capability, preemption vulnerability.
- All simultaneously active bearers by one UE are provided are provided by the same P-GW.

3.1.3.4 *GPRS Fundamentals*

The packet switched domain of an UMTS network is an evolution of GPRS and thus closely related to it. First defined by the 3GPP in Release 99, it focuses its improvements over GSM mostly on the radio aspects, while keeping the core network GPRS architecture intact at large. 3GPP TS 23.060 [3GP13c] defines the basic aspects involving GPRS protocols and its system architecture. TS 29.060 [3GP13b] describes the specifics of GTP flowing across the Gn and Gp interfaces which forms the foundation for our work.

As shown in Figure 3.1, user traffic originating at any MS connected to the radio network flows through a Node B (also called base station), which provides radio connectivity. Multiple Node B are aggregated into a RNC. The base stations and RNCs form the UTRAN, which is typically connected by back-haul fiber links to the core network part formed by the SGSN and the GGSN.

One role of the SGSN is to serve as mobility anchor for mobile devices. It is also the endpoint for RRC-based signaling and the RAB, the radio counterpart to the core network user traffic tunnel. The GGSN provides the gateway to the public Internet. The Gn interface connects those two nodes, using the GTP protocol to encapsulate user as well as control plane traffic as seen in the protocol stack in Figure 3.4. GTP is further separated into GTP-C, facilitating control message exchange, and GTP-U for transporting user traffic through tunnels in the core.

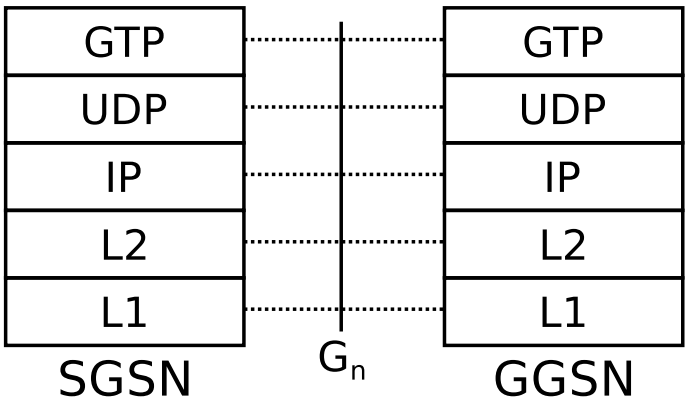


Figure 3.4: Typical signaling protocol stack at the Gn interface between SGSN and GGSN.

Before diving into specifics of GTP messaging, we give a short overview on the packet switched domain of an UMTS network. This domain is closely related to the GPRS

part introduced for GSM. UMTS, first defined by the 3GPP in Release 99, focuses its improvements over GSM mostly on the radio aspects, while keeping the core network GPRS architecture intact at large. 3GPP TS 23.060 [3GP13c] defines the basic aspects involving GPRS protocols and its system architecture. TS 29.060 [3GP13b] describes the specifics of GTP flowing across the Gn and Gp interfaces which forms the basis for our work.

As shown in Figure 3.1, user traffic originating at any MS connected to the radio network flows through one of the Node Bs, providing radio connectivity. Multiple Node Bs are aggregated by a RNC. Node Bs and RNCs form the UTRAN, which is typically connected by back-haul fiber links to the core network part formed by the SGSN and the GGSN.

One role of the SGSN is as the mobility anchor for mobile devices, and it is the endpoint for RRC-based signaling and the RAB. The GGSN provides the gateway to the public Internet. The Gn interface connects those two nodes, using the GTP protocol to exchange user as well as control plane traffic as seen in the protocol stack in Figure 3.4. GTP is further separated into GTP-C, facilitating control message exchange, and GTP-U for transporting user traffic through tunnels.

Table 3.1: GTP header format (TODO: which one exactly. This looks like v2-U but should actually better be v1!)

	Bits							
Octets	8	7	6	5	4	3	2	1
1	Version			P	T	Spare	Spare	Spare
2	Message Type							
3	Message Length (1st Octet)							
4	Message Length (2nd Octet)							
m to k(m+3)	If T flag is set to 1, then TEID shall be placed into octets 5-8. Otherwise, TEID field is not present at all.							
n to (n+2)	Sequence Number							
(n+3)	Spare							

Table 3.1 lists the standard GTPv1 header format used in any GPRS and 3G network.

Table 3.2 lists the default header format for a GTPv2-C message used in LTE networks.

request/response messaging response types: Possible context response types and which request types they answer:

- 192: “non-existent” UPDATE & DELETE ONLY
- 193: “invalid message format” UPDATE & DELETE ONLY
- 199: “no resources available” CREATE ONLY anywhere in the network to allocate context
- 200: “service not supported” UPDATE ONLY
- 201: “mandatory IE incorrect”
- 202: “mandatory IE missing”

Table 3.2: 12 Byte GTPv2-C header format.

	Bits							
Octets	8	7	6	5	4	3	2	1
1	Version			P	T=1	Spare	Spare	Spare
2	Message Type							
3	Message Length (1st Octet)							
4	Message Length (2nd Octet)							
5	Tunnel Endpoint Identifier (1st Octet)							
6	Tunnel Endpoint Identifier (2nd Octet)							
7	Tunnel Endpoint Identifier (3rd Octet)							
8	Tunnel Endpoint Identifier (4th Octet)							
9	Sequence Number (1st Octet)							
10	Sequence Number (2nd Octet)							
11	Sequence Number (3rd Octet)							
12	Spare							

- 204: “system failure” CREATE & UPDATE ONLY
- 209: “user authentication failed” CREATE ONLY rejected for various reasons

Information Element (IE) calculation basis

GTP Header: 12 Byte IE header and footer: 2 Byte Maximum minimum data size including all IEs: 221 Byte + 12 Byte Header + 2*37 Extension Header = 307 Byte Minium size of mesage with just mandatory IEs: 12 + 30 + 2*5 = 52 Byte

307 Bytes: calculation from our dataset Total maximum signaling traffic with this calculation: 117.15GB Ratio: 0.10% 52 Bytes: Total maximum signaling traffic with this calculation: 19.84GB Ratio: 0.02% Total traffic: 122758578593993

Tunnels state is held in the SGSN and GGSN as PDP Context data structures. These contain various information, such as the device IP address, International Mobile Subscriber Identity (IMSI), and a tunnel identifier. The concept is used to isolate user traffic from core network control plane signaling and to provide certain QoS guarantees to the user traffic. Multiple QoS profiles per device can also be established by setting up up to ten secondary contexts beyond the primary PDP context. However, QoS secondary contexts are very rarely in use today, any user-plane IP traffic is typically transported within the primary “best effort” tunnel.

The GTP-C signaling, responsible for the context management interactions, contains procedures for managing data paths, MS locations, mobility, and, of course, tunnels. GTP messages usually come as request-response pairs. Neither part has fixed size, but is rather constructed from a number of IEs, many of which are either optional or variable length through additional optional fields.

The focus of our work will be the three Tunnel Management message pairs involved in the maintenance of PDP Contexts. These are:

- The **Create Context Message**, which is part of several larger control procedures that activate the GTP tunnel for a mobile device. These can be initiated from the

network as well as the device itself, again depending on the specific implementation of the architecture. When a GGSN receives this request from an SGSN, it attempts to complete the Context creation. Depending on the outcome, a response is sent back, indicating the success or failure of the operation. Typical failures include failed user authentication, lack of resource, or unrecoverable system failures.

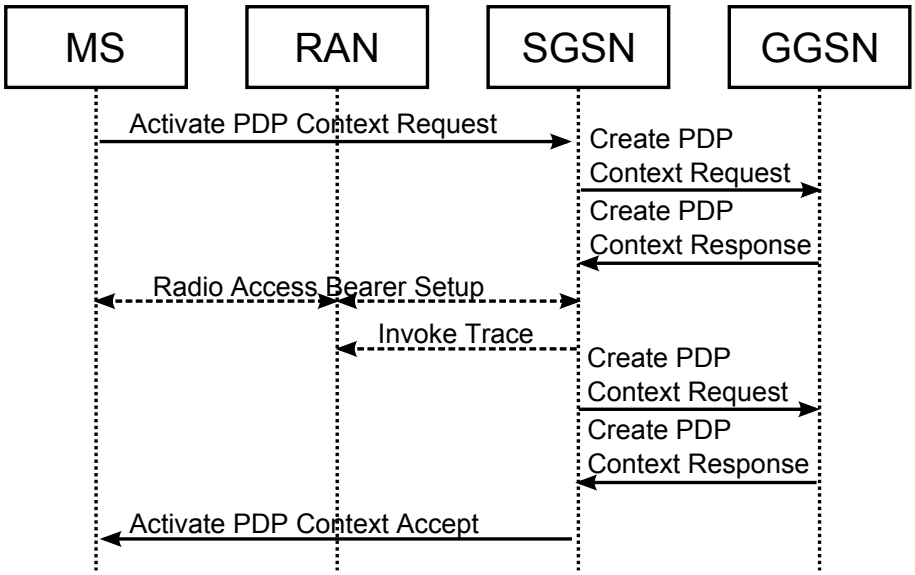


Figure 3.5: PDP Context Activation Procedure in a UMTS network.

Figure 3.5 shows the *PDP Context Activation Procedure* as defined in [3GP13c]. Some additional CAMEL procedures may be involved in the creation but are not of interest in the context of this paper as they are not required. Tunneling messages are usually triggered by other procedures on different interfaces. In the case depicted here, the procedure is initiated by the mobile device through a RANAP protocol *Activate PDP Context Request* typically sent when establishing a mobile data connection.

Generally speaking, any Create Request is part of a *GPRS PDP Context Activation* procedure, which can happen under several circumstances, the aforementioned one being the typical, but also during each *Secondary PDP context Activation* procedure for every tunnel beyond the first. When a GGSN receives this request from an SGSN, it attempts to complete the Context creation. Depending on the outcome, a response is sent back, indicating the success or failure of the operation. Typical failure codes observed in our measurements were either due to incorrect information supplied by the device ("user authentication failed"), due to malformed messages (e.g. "invalid message format"), or indicated problems or temporary overload in the network ("no resources available" and "system failure").

- **Delete Context Message**; This indicates the immediate release of the Context involved. Together with the Create event these mark the beginning and the end of every GTP tunnel, making them good candidates to determine tunnel durations for our load evaluations.

The third type of Tunnel Management messages are the *Delete Context Request* and *Response*, indicating the immediate release of the Context involved. They are part of

  - The *GPRS Detach* procedure from the SGSN to the GGSN, when a device completely deactivates its data services.
  - The *GPRS PDP Context Deactivation* procedure from the SGSN to the GGSN, if only one specific tunnel is to be removed.
  - The *part of PDP Context Deactivation Initiated by GGSN* procedure signaled to the SGSN.

- **Update Context Messages**; Several procedures also emit tunnel update messages, when some aspect of the tunnel has changed, e.g. occurring in mobility and load-balancing related procedures but also procedures involving secondary tunnels for a device. By observing Update Context message one could, for example, capture most forms of mobility happening in the network, and get a good picture of correlations between mobility and tunneling characteristics.

  The possible causes for an *Update Context Request* are as following.

  - The mobile devices moves between SGSNs, causing a *GPRS inter-SGSN Routing Area Update* procedure.
  - Parameters belonging to the context such as the assigned QoS are altered using the the *PDP Context Modification*.
  - As part of *Context redistribution and load balancing* procedures.
  - The MS switches between UMTS and GPRS access technologies, causing a *Inter-system intra-SGSN Update* procedure. Note that the same tunnel can be used regardless of the radio technology.
  - As part of a direct RNC to GGSN GTP-U tunnel activation procedure, thereby circumventing the SGSN. Or, finally,
  - To activate secondary PDP contexts using the *Secondary PDP Context Activation* as previously described.

  By observing Update Context message one could, for example, capture most forms of mobility happening in the network, and get a good picture of correlations between mobility and tunneling characteristics. Additionally, tunnels using UMTS and GPRS radio technology can be distinguished, which should in theory lead to wholly different pictures, as nowadays GSM/GPRS is either used in older models or feature phones, or in mobile scenarios in rural areas where the larger GSM cells are more prevalent. Both could indicate that the data session will be rather short due to either clumsy devices or the low throughput rates of GPRS.

The variable-length nature of these messages makes evaluating the imposed network signaling overhead rather difficult. For example, the Create Context Response consists of up to 36 IEs, some of them mandatory, most either conditional or optional. Including the headers of both the packet and the individual elements, the minimum size (counting

only the required bytes of variable length elements) is 52 bytes, while the lower bound for the message size with all IEs present is 307 bytes.

Taking the maximum size we arrive at a naive estimate of the maximum overhead on user traffic induced by tunnel management signaling in our dataset. The estimated ratio of (tunnel management) signaling traffic to total user plane traffic in our dataset is a minute 0.10%. Therefore, the volume of control plane traffic appears to be non-critical in this setup. Thus, we assume that the overload problems mentioned above arise rather in areas affected by signaling except for the pure transport of data, such as the memory profile of the states kept in the gateway nodes, the time required to process the large number of information held in the messages, or the imposed latency through several message round trips during transactions.

Tunnels are defined in the SGSN and GGSN in PDP Context data structures. These hold various information related to a tunnel, such as the device IP address, IMSI, and a tunnel identifier. A tunneling concept is used for user traffic to isolate it from core network control plane traffic and to provide certain QoS guarantees to the user traffic. To distinguish multiple QoS profiles per device, up to ten additional secondary contexts can be established beyond the primary PDP context, all with different QoS allocations. However, secondary contexts are rarely in use today, and any user-plane IP traffic is transported within the primary "best effort" tunnel.

As already mentioned, GTP-C signaling is used to administer these contexts. Across the Gn path, it contains procedures for managing data paths, MS locations, mobility, and, of course, tunnels. We take a specific look at the last one. GTP messages usually come as request-response pairs. Neither part has fixed size, but is rather constructed from a number of IEs of partially variable length.

The focus of our work will be the three Tunnel Management message pairs involved in the maintenance of PDP Contexts. These are the *Create, Update,* and *Delete PDP Context Requests* and *Responses*. Each pair, including their causes and possible effects, will be treated in a separate section, with the Create and Delete messages forming the substrate for our investigations presented in this paper.

The variable-length nature of these messages makes evaluating the imposed network signaling load rather difficult. For example, the Create Context Response consists of up to 36 IEs, some of them mandatory, most either conditional or optional. Including the headers of both the packet and the individual elements, the minimum size (counting only the required bytes of variable length elements) is 52 bytes, while the minimal maximum size with all IEs present is 307 bytes.

Taking this maximum value we arrive at a naive estimate of the maximum overhead on user traffic imposed by tunnel management signaling in our dataset. The ratio of (tunnel management) signaling traffic to total user plane traffic is a minute 0.10%. Therefore, the sheer volume of control plane traffic appears to be non-critical in this setup. We assume thus that the overload problems mentioned above arise rather in areas affected by signaling except for the pure transport of data, such as the memory profile of the states kept in the gateway nodes, the time required to process the large number of information held in the messages, or the imposed latency through several message round trips during transactions. The detailed mechanics of system load could be a field of investigation for future work.

linked NSAPI: indicates the NSAPI assigned to any one of the already activated PDP contexts for this address/phone ("foreign key"?)

GTP INFLUENCING STATE MACHINES As indicated, most nodes in a cellular mobile network keep all sorts of state characterizing the data connection. For the tunnel management aspects, two state machines are of special note, namely the Mobility Management and RRC state machines. The former, defined in [3GP13c], describes the general state of the data connection, and switches states based either on an idle timer, or when new packets arrive for the mobile device. The RRC state machine depicted in Figure 3.6 governs the usage of radio channels. State changes happen again depending on user activity and inactivity. Based on the state both procedures can enable and disable radio tunnels as well as core network tunnels, making them a good example of user traffic dynamics directly influencing core network signaling, similar to the observations in [LBW07].

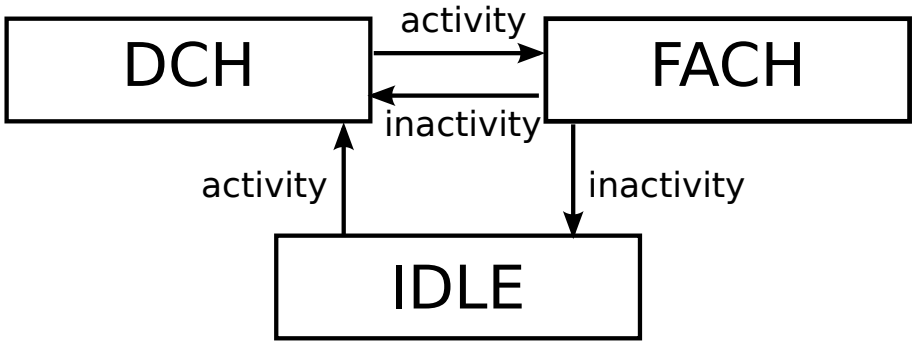


Figure 3.6: Simplified RRC State Model.

As indicated before, most nodes in a cellular mobile network keep all sorts of states characterizing the data connection. For the tunnel management aspects, two state machines are of special note.

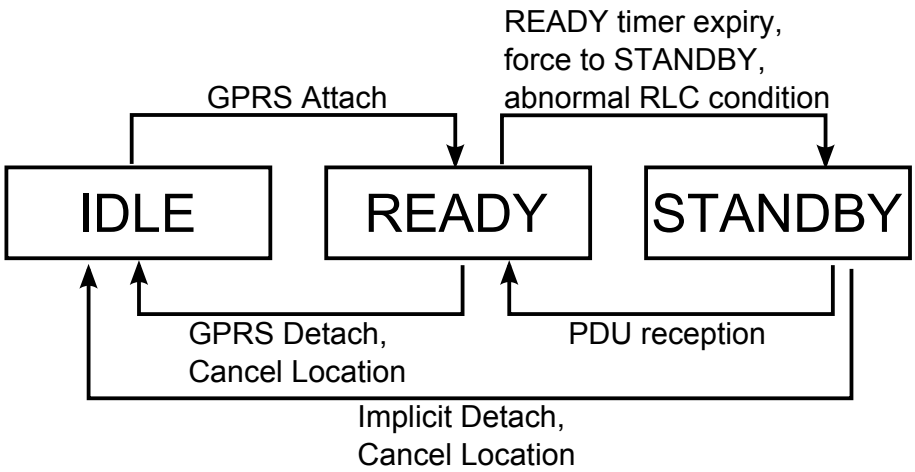


Figure 3.7: SGSN Mobility Management State Model.

First, consider the Mobility Management state machine depicted in Figure 3.7, defined in [3GP13c], and held in both the SGSN as well as the mobile device. It describes the general state of the data connection, and switches states based either on an idle timer, or when new packets arrive for the mobile device. Therefore, it also controls tunnel management, as the involved GPRS Detach and Attach procedures involve deleting and

creating contexts. We identify user traffic dynamics as one vector to influence core network signaling, similar to the observations in [LBW07].

The RRC state machine shown in Figure 3.6 governs the usage of radio channels, i.e. spectral and temporal usage of the wireless interface. State changes happen depending on the inter-arrival time of user packets. In this case, if the state machine transitions to the IDLE state, the RAB on the path between mobile device and SGSN is not needed anymore and will be deleted, in most cases destroying the SGSN and GGSN PDP Context as well.

per Public Land Mobile Network (PLMN) node, cf. 3GPP TS 23.401 clause 5.7.

DISCUSSION OF GTP SIGNALING Looking at the Create, Update and Delete PDP Context Request and Reply message pairs we can already directly deduce some possibly load-related information. The total tunnel duration originates from the time delta between corresponding Create and Delete events. The shorter those tunnels are the higher will be volume of signaling messages and the necessary processing for these messages. Conversely, longer tunnel durations cause an increased overall memory footprint in the involved nodes to store the PDP Contexts. Large numbers of update messages, especially combined with frequent Radio Access Technology (RAT) switches, are usually an indicator for highly mobile devices switching their routing area. The time between a request and its corresponding response could also be an indicator for the amount of processing involved for this message as well as the current general processing load at the GGSN.

As discussed, most of the actions in the network as well as in the mobile devices are reflected in the presented tunnel management messaging. Therefore, taking a look at the dynamics of this control aspect in real networks gives valuable insights on the influence of many of the networks' aspects.

Looking at the Create, Update and Delete PDP Context Request and Reply message pairs we can already deduce a certain amount of information. Measuring the time delta between corresponding Create and Delete events obviously gives the total duration a tunnel was established. Given the amount of user-plane IP traffic transferred, when the tunnel durations are short, we can expect the number of tunnel creation and deletion events to go up instead, resulting in a higher volume of signaling messages and an increase in processing for these messages. Conversely, longer tunnel durations cause an increased overall memory footprint in the involved nodes to store the PDP Contexts. Large numbers of update messages, especially combined with frequent RAT switches, are usually an indicator for highly mobile devices switching their routing area. This mobility behavior could be investigated by evaluating the update messages.

As discussed, most of the actions in the network as well as in the mobile devices are reflected in the presented tunnel management messaging. Therefore, taking a look at the dynamics of this control aspect in real networks gives valuable insights on the influence of many of the networks' aspects.

- Cellular data network infrastructure characterization and implication on mobile content placement Xu[Xu+11]

- Regarding investigations of network infrastructure we again see an user network layer centric approach in [Xu+11]

- nur in die darwin section: Traffic monitoring and analysis in 3G networks: lessons learned from the METAWIN project [Ric+06]
- A Comparison Between One-way Delays in Operating HSPA and LTE Networks [Lan+12]
- Discovering Parameter Setting in 3G Networks via Active Measurements [BRB08]
- A first look at cellular machine-to-machine traffic: large scale measurement and characterization [Sha+12a]
- Comparative Performance Study of LTE Downlink Schedulers [BT13]
- Source traffic modeling of wireless applications [SLT00]
- Traffic analysis at short time-scales: an empirical case study from a 3G cellular network [RHR08]
- Traffic modeling and characterization for UMTS networks [KLL01]
- Study on non-MTC mobile data applications impacts [3GP11b]
- Two parallel approaches to network data analysis [Bae+11]
- Statistical inference for exploratory data analysis and model diagnostics [Buj+09]
- 23.843 [3GP13j] Study on Core Network (CN) overload solutions
- 22.801 [3GP11b] Study on non-MTC mobile data applications impacts; for angry birds // relwork important
- 23.843 [3GP13j] Study on Core Network (CN) overload solutions
- 24.826 [3GP11a] Study on impacts on signalling between User Equipment (UE) and core network from energy saving; deals mostly with switching off cells and moving over UEs, not actual core network efficiency
- 29.807 [3GP13k] Study on GTP-C overload control mechanisms

Correlation to stories about carrier complaints over (radio) "signalling storms" and 3gpp R8 Fast Dormancy [3GP12f] and [GSM12]

3.2 RELATED WORK

Existing research can roughly be divided into two areas. First are attempts to infer control plane behavior through application layer active measurement at the mobile device or through synthetic traces or traces from other radio networks (e.g. 802.11). Second, investigations of user traffic characteristics by means of real 3G core network traces. This paper does not strictly fall in either of these two categories but instead aims to provide insights into the control plane from the perspective of the core network. It is also an extension to our Research Report[Met+12] aiming to provide more in-depth statistical analyses to the control plane. However, both areas are still highly relevant at related to our work. Therefore we present a selection of publications from these fields and detail the interesting aspects for this work.

3.2.1 *Device Active Measurement Investigations*

Recently, stories about signaling storms and overloaded control planes in mobile networks reached popular news media [Cor11; Don12]. These stories blame a specific combination of mobile device type, operating system and application to cause excessive amounts of signaling in the radio network. The Android version of the popular casual game "Angry Birds" is a free download, and uses regularly refreshed advertisements displaying after every game stage. Now imagine a large amount of devices setting up and tearing down data connections only to retrieve new ads and therefore causing tens of control plane messages on each retrieval, which could strain the signaling-heavy structure of current networks.

The dynamics behind such events are already under investigation by several publications, focusing on the impact at the radio interface and on RRC signaling but paying little attention to potential aspects in the core network. A paper on cross-layer interaction in mobile cellular networks falls into this category [Qia+11], discussing interaction, e.g., between the application layer and the RRC (such as seen in the "Angry Birds" case) and its consequences for device energy consumption and radio channel allocation efficiency. The authors argue that there is much room for improvement in this area, and propose some enhancements.

In [LBW07], mobile network traces are used to simulate a malicious signaling storm by transmitting low-volume user plane traffic with inter-departure times slightly larger than the transition timers in the RRC state machines. This constantly causes signaling to occur. The authors propose tools to detect this, and discuss a possible scale of this type of denial-of-service attack.

Wang et al.[Wan+11b] developed NetPiculet to probe mobile networks for middle boxes that alter traffic and affect performance, e.g. NAT, firewalls, or non-transparent proxies. Such nodes were present in a large portion of the investigated networks, increasing device power usage and download durations while providing themselves a surface for certain attacks.

Looking at the transition of RRC states, which is briefly explained in Sec. TODO, we find in [Per+09] some simple albeit effective application layer methods at the mobile device to investigate these transitions. This is further enhanced by research from Schwartz et al.[Sch+13] using this technique to analyze the radio signaling load and thus power efficiency from different applications including the aforementioned

3.2.2 *Research Based On Core Traces*

As stated, all of these approaches cannot take the core network into account, as they do not have access to the necessary measurement infrastructure. The following research papers all have some kind of core network dataset. Most of them do not directly tackle signaling, however.

The authors of [Sha+11] and [Pau+11] both take the approach of looking at high-level user traffic characteristics in a mobile network, focusing on temporal and spatial variations of user traffic volume and peeking at the influence of different devices on this metric. Additional user flow and session traffic metrics are being looked at in [ZÅ12] with the conclusion that, in comparison to wired traffic, much more shorter flows are

occurring. If this shorter-but-more theme is also evident in signaling traffic, this could translate into an increased signaling load.

In 2006, Svoboda et al. [Svo+06] conducted a core network measurement study of various user traffic related patterns, and also provided an initial insight into PDP context activity and durations. Another recent publication at [He+12] provides an investigation aimed at RRC signaling on the RNC to SGSN link but not at GTP signaling at the SGSN to GGSN path which we deem more important for our core network load characteristics research. The authors classify their evaluations based on device model and vendor and on the application type, and find that different devices have strongly different RRC characteristics, which could possibly also have an impact on GTP signaling. Here the RRC evaluation was done in a direct manner using explicit logs from the RNC. A 2010 publication[Qia+10] however uses the indirect RRC inferring method described earlier on a core network TCP trace data set and finds that the involved RRC state machine is largely inefficient in terms of signaling overhead and energy consumption for typical traffic patterns seen in the data.

The authors of [RRCpt] give us some thoughts on the influence of core network elements on one-way delays in mobile networks, also providing us with initial clues on the expected load impact of these elements for our own investigation. A final paper [RCD10] presents some Denial of Service (DoS) attack scenarios on these networks from a theoretical view. As a DoS either needs to find a weak (performance-wise) link in an architecture or a good source for an amplification attack – small information causes a large amount of information to be computed or transmitted – this is also very helpful information in evaluating core network load and finding bottlenecks.

All of these touch to some degree parts of the areas tackled in this paper, but we think that the combination of the focus on core signaling, a statistical evaluation of PDP Contexts with an investigation of sources influencing these, and a simple load model are genuine contributions of our work.

This work is a continuation of our previous evaluations conducted in [Met+12] and [MRR13]. However, to our knowledge there are no other directly related predecessors in literature as this paper provides a novel model. However, some efforts have been made to investigate the special properties of mobile networks and its traffic. These include attempts to infer control plane behavior through active measurements at the mobile device or synthetic traces and investigations of user traffic characteristics by means of real 3G core network traces. The authors of [Qia+11] discuss cross-layer interactions in mobile cellular networks and the consequences for device energy consumption and radio channel allocation efficiency. In [LBW07], mobile network traces are used to simulate a malicious signaling storm by transmitting low-volume user plane traffic with specially crafted inter-departure times, causing signaling to occur constantly. Looking at the multitude of radio network control state machines, we find in [Per+09] some simple yet effective application layer methods investigating transitions of these state machines. This is further elaborated on by [Sch+13], using this technique to analyze the radio signaling load and thus power efficiency from several different applications.

Having access to core network datasets, the authors of [Sha+11] and [Pau+11] both take the approach of looking at high-level user traffic characteristics, focusing on temporal and spatial variations of user traffic volume and investigating the influence of different devices on this metric. Additional user flow and session traffic metrics are being studied at in [ZÅ12] with the conclusion that, in comparison to wired traffic, much

more shorter flows are occurring. In 2006, a core network measurement study of various user traffic related patterns was conducted [Svo+06], providing an initial insight into PDP context activity and durations. [He+12] provides an investigation aimed at radio network signaling. The authors find that different devices have also different signaling characteristics. A 2010 publication [Qia+10] indirectly infers radio signaling from TCP traces concluding that very commonly occurring traffic patterns cause large signaling overhead and high energy consumption. The authors of [RRCpt] give some thoughts on the influence of core network elements on one-way delays in mobile networks, hinting the expected load impact of these elements.

3.3 DATASET AND METHODOLOGY

3.3.1 *Network and Monitoring Setup*

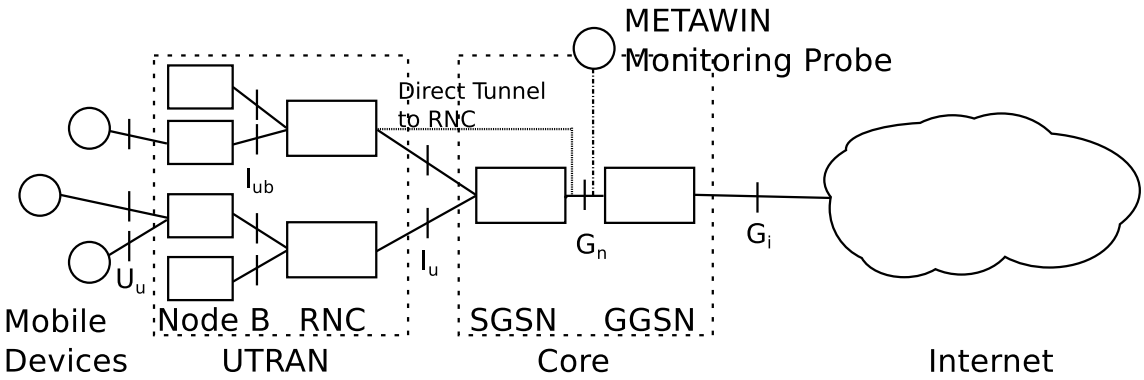


Figure 3.8: Location of the METAWIN monitoring probe in the 3G core network

For our analysis, we use the Measurement and Traffic Analysis in Wireless Networks (METAWIN) monitoring system developed in a previous research project and deployed in the network of an Austrian mobile operator. More in-depth information about METAWIN is available in [Ric].

The location of the measurement probe within the core network is highlighted in Figure 3.8.

The location of the measurement probe at the Gn interface within the core network, marked in Figure 3.8, gives access to both wide-area mobility signaling (not analyzed in this paper) and signaling related to user-plane IP traffic (which we want to scrutinize). The METAWIN monitoring system extracts and correlates information from the lower layers of the 3GPP protocol stack, and specifically the GTP protocol on the Gn interface [3GP13b]. This includes the Radio Access Technology (RAT) identifier as well as the terminal types of the mobile clients. The latter is determinable by the Type Allocation Code (TAC) part of the International Mobile Equipment Identity (IMEI) (cf. [3GP13h]) and will be discussed later in detail.

To meet privacy requirements, the METAWIN system anonymizes captured data on the fly at multiple layers: the application-level payload is removed and all user identifiers (e.g. IMSI) are hashed with a one-way function before recording. That is, single MSs in our dataset may be differentiated by means of an anonymized Mobile Station Identifier (MS-ID), but not traced back to the actual customer. The packet capturing

hardware deployed within the METAWIN monitoring system is synchronized using Global Positioning System (GPS). Accordingly, the packet timestamps have an accuracy of ±100 ns or better [Dono2, p.97-98].

As said before, the GGSN acts as the IP-layer gateway for the user traffic. It is in charge of setting up, maintaining, and tearing down a logical connection to each active MS. This logical connection, the PDP context, is conceptually similar to a dial-up connection. During set up, an IP address is dynamically assigned to the MS. In the network under study, a so-called *direct tunnel* setup might be used for MSs connected via UMTS/HSPA. Such a setup consists of a direct link between GGSNs and the RNCs which is used for transporting user-plane data traffic only. Since signaling procedures such as mobility management are carried out by the SGSNs, a GGSN always has to send signaling packets via the path involving the SGSNs. Monitoring the Gn interface thus gives us access to both wide-area mobility signaling (not analyzed in this paper) and signaling related to user-plane IP traffic (which we want to scrutinize). For more information on the 3G network structure, please refer to [BMC04].

The METAWIN monitoring system extracts and correlates information from the lower layers of the 3GPP protocol stack, and specifically the GTP protocol on the Gn interface [3GP13b]. This includes the Radio Access Technology (RAT) identifier as well as the terminal types of the mobile clients. The latter is determinable by the Type Allocation Code (TAC) part of the International Mobile Equipment Identity (IMEI) (cf. [3GP13h]) and will be discussed later in detail.

To meet privacy requirements, the METAWIN system anonymizes captured data on the fly at multiple layers: the application-level payload is removed and all user identifiers (e.g. IMSI) are hashed with a one-way function before recording. That is, single MSs in our dataset may be differentiated by means of an anonymized MS-ID, but not traced back to the actual customer. The packet capturing hardware deployed within the METAWIN monitoring system is synchronized using GPS. Accordingly, the packet timestamps have an accuracy of ±100 ns or better [Dono2, p.97-98].

3.3.2 *Dataset Description*

The METAWIN-recorded dataset used in our evaluation is a week-long trace from the third week of April 2011. It consists of 2.2 billion aggregated flows for the user traffic and 410 million GTP Tunnel Management transactions, the latter representing the data base for this paper. It was tapped at one of the GGSNs of the operator, and contains about half of the total traffic volume handled by the operator in this period. The GTP data contains the response codes for each transactions. With these codes, failed interactions can be sorted out and treated separately.

We fed the records into a SQL database, and conducted further evaluations through scripted queries on the database. Any privacy-relevant data, e.g. the International Mobile Equipment Identity (IMEI), MS-ID and any IP address involved, is only visible as hashes and is processed in a privacy-preserving manner. Since the hashing of the IMEI is consistent throughout the dataset, user traffic flows and the GTP data can be cross-correlated despite anonymization, giving the opportunity for further research.

In order to evaluate our models, we use data gathered from a nation-wide mobile operator. This allows for precise core network evaluations and the creation statistical

fits for the observed processes. In this section we first describe the dataset used for the evaluation and afterwards we derive the random variables required for our models.

All data was collected by the METAWIN monitoring system [Ric] with measurement probes located at the Gn interface within the core network, enabling broad access to signaling. For this investigation we exclusively use GTP protocol data gathered by METAWIN. This includes the RAT identifier as well as the terminal types of the mobile clients, by use of the Type Allocation Code (TAC) part of the IMEI. To meet privacy requirements, METAWIN anonymizes all captured data. The application-level payload is removed and all user identifiers are hashed with one-way functions before recording. Individual MSs in our dataset can be differentiated by the hashed MS-ID, but not traced back to the actual user.

The dataset used in our evaluation is a week-long trace from the third week of April 2011. It consists of 2.2 billion aggregated flows for the user traffic and 410 million GTP Tunnel Management transactions. It was tapped at one of the GGSNs of the operator, and contains about half of the total traffic volume handled by the operator in this period.

Our dataset, kindly provided by a nation-wide mobile operator and recorded using the METAWIN monitoring system, was taken in the third week of April 2011. It consists of seven days of aggregated flow-level data for the user traffic and a summary entry for every GTP Tunnel Management transaction, the latter representing the data base for this paper. It was tapped at one of the GGSNs of the operator, and contains about half of the total traffic volume handled by the operator in this period. The GTP data contain the response codes for each transactions. With these codes, failed interactions can be sorted out and treated separately.

We fed the records into a SQL database, and conducted further evaluations through scripted queries on the database. Any privacy-relevant data, e.g. the IMEI, MS-ID and any IP address involved, is only visible as hashes and can only be processed in anonymized form. Individual device types can be identified in form of the unhashed TAC on every entry. Since the hashing of the IMEI is consistent, user traffic flows and the GTP data can be cross-correlated despite anonymization, giving the opportunity for further research.

3.3.3 *Device Identification*

Individual device types can also still be identified in form of the TAC on every entry. The TAC is part of the IMEI, uniquely identifying each device type [3GP13h]. The rest of the IMEI constitutes the serial number of the involved devices, which is not present in the data.

In our dataset, the TAC field is provided in cleartext, whereas the IMEI is only available in hashed form to preserve the privacy of device owners. The TAC is contained in the first eight decimal digits of the IMEI, uniquely identifying each device type [3GP13h]. The rest of the IMEI constitutes the serial number of the involved devices.

TACs are managed by the GSM Association (GSMA) which in turn assigns local organizations, distinguished by the first two digits of the TAC as Reporting Body Identifier, to allocate TACs to manufacturers. However, this allocation information is not freely available. Commercial databases exist, but this is neither affordable for research institutions, nor is it conducive to our goal of providing information to the public. While there are some websites that allow one to query for specific TACs for non-commercial

purposes, only very few efforts attempt to collect TAC information into a publicly available database. We based our data-mining efforts on a publicly available set[5], with some additional devices collected on our own. Since the unit identification part of the IMEI is just six decimal digits long, popular devices will even be assigned more than one TAC, making the acquisition of all relevant TACs even more complicated.

For our investigation, we went through large portions of the TACs present in our dataset, and identified and categorized the most important entries. In this case, importance means various metrics like the traffic volume, the number of flows, and the number of GTP signaling messages for each TAC.

3.3.4 *Device Classification*

For our investigation, we went through large portions of the TACs present in our dataset, and identified and categorized the most important entries. In this case, importance means various metrics like the traffic volume, the number of flows, and the number of GTP signaling messages for each TAC.

After having available the device names for most TACs, we were able to add meta-information and categorize the entries based on their device type and operating system. For the device type we partitioned the devices roughly into smartphones, regular mobile phones and 3G USB dongles or 3G/WiFi routers. The operating system includes most of the popular incarnations found in the network at measurement time, including Android, iOS, and Symbian. Note however that many devices, especially USB dongles cannot be linked to any specific OS.

After having available the device names for most TACs, we were able to add meta-information to the entries in form of the following categories:

- The device type. We distinguished between smartphones, regular mobile phones and feature phones, and 3G USB dongles or 3G/WiFi routers.
- The operating system of the device (if known), such as Android, iOS, Series 40, BlackBerry OS etc. This is especially interesting to identify potential differences in the core network signaling patterns of devices. Note however that we cannot link USB dongles and OS types from the TAC.

[TODO: extend this section]

3.3.5 *TAC Statistics and Evaluation Validity*

It is important to know whether our TAC mappings provide sufficient useful data to allow for the envisioned device discriminating statistics. Therefore, Table 3.4 provides some statistics on our knowledge of devices in the dataset. About 80 percent of all distinct devices active could be identified. Looking at the total number of GTP signaling messages, we see that we can determine the device name of over 90 percent. The flow data shows an even clearer picture, as we can identify almost all of the devices involved.

After applying the categorization to the TACs we evaluate the device composition in the network. The two largest portions of devices are smartphones and 3G dongles, while classic cell phones do not seem to play a major role anymore.

5 http://www.mulliner.org/tacdb/, Mulliner, C.

Initially, one planned endeavor was to investigate possible peculiarities of business phone behavior, especially of those easily identifiable Blackberry OS phones, but the number of distinct Blackberry devices in the dataset is too low to draw conclusions of any significance.

One observation across all device types is that about 18 percent of all mobile devices have activated their mobile data service and have signaling traffic, but do not cause any use plane traffic.

The difference between 3G dongles and smartphones is also noteworthy. While the former cause large amounts of user plane traffic (compared to the device numbers), they are responsible for but a low number of core network signaling events and tunnels. This picture is reversed for smartphones.

As we are working with an incomplete TAC database it is important to know whether our TAC mappings provide sufficiently useful data to allow for the envisioned device discriminating statistics. Therefore, Table 3.4 provides some statistics on our knowledge of devices in the dataset. About 80 percent of all distinct and active devices could be identified. Looking at the total number of GTP signaling messages, we see that we can determine the device name of over 90 percent. The flow data shows an even clearer picture, as we can identify almost all of the devices involved.

After applying the categorization to the TACs we evaluate the device composition in the network. The two largest portions of devices are smartphones and 3G dongles, while classic cell phones do not seem to play a major role in the packet-switched domain anymore. One observation across all device types is that about 14 percent of all mobile devices have activated their mobile data service and have signaling traffic, but do not cause any user plane traffic.

The difference between 3G dongles and smartphones is also noteworthy. While the former cause large amounts of user plane traffic (compared to the device numbers), they are responsible for but a few core network signaling events and tunnels. This picture is reversed for smartphones.

3.4 MOBILE CORE NETWORK LOAD DEFINITION

work in 23.843 [3GP13j] Study on Core Network (CN) overload solutions GTP-C retransmission of unacknowledged requests" currently: semi-static DNS based load balancing (does this apply only to LTE/SGW?)

Before beginning the evaluation, the primary question driving this investigation was: “How can load in a core network be defined and measured?” A summary of our thoughts to this question follows here.

With the basics of the architecture in mind, a top candidate for high load is the GGSN. All traffic leaving or entering the packet switched domain must go through this element, and it is in control of the described GTP signaling procedures as well. Being an endpoint for the GTP tunnel makes it responsible to sort and encapsulate incoming traffic into the corresponding user tunnel. To accomplish this a lot of state has to be kept – and processed when signaling occurs. Therefore, our working hypothesis is, that in order to determine load the GGSN needs to be monitored closely and any traffic related to this node investigated for indications of the current load.

For our definition of the term “load” we differentiate between signaling load and overhead on the one hand and processing load and memory consumption on the other

hand. Both are measures of load at specific nodes. While the former mostly has an impact on the actual network traffic, the latter can only be grasped inside the network element. With our data we can directly investigate the signaling traffic but indirect measures for the processing load and memory usage have to be found. In the rest of this section we evaluate the results of several approaches to both of these definitions of load.

While looking at the GGSN may be the most obvious choice, it is by far not the only one. In addition to GTP tunnels the SGSN has to handle RAB and mobility management as well. However, it is assumed, that there are more regionally distributed SGSN nodes present in a typical mobile network. This means that a single element would have to handle less mobile devices and therefore load. One has also to bear in mind that the SGSN can be completely circumvented by setting up a direct tunnel between GGSN and RNC.

Apart from the two gateways directly inside the traffic path there are several other nodes essential to the control plane decision making, which may very well be also very load-sensitive. The HLR for example is a central database storing all user related information which need to be retrieved any time a user needs to undergo initial authentication and authorization. Typically, the procedures the elements are involved in are fewer and they are also harder to investigate with the data available to us. Hence, it was decided to concentrate just on the case of the GGSN.

3.4.1 *Load Influencing Factors*

Having described our understanding of core network load we can now move to discuss some of the factors that could influence the load, making them targets for our evaluation.

The first and arguably one of the most important factors are the mobile devices themselves. Specifically, this covers the behavior of the network layer 1 and 2 implementation (sometimes called "'baseband") as well as the Operating System (OS) and the running applications. The OS and baseband decide when the device should establish a mobile data connection, how long the connection is held, or which mobile technology takes preference. Depending on the access technology, be it GPRS, EDGE, UMTS, HSPA, or HSPA+, we can expect subtle differences through their specifications, e.g. in the timing of the radio transmission intervals, which could influence our investigation.

Some specific tunnel duration properties could stem from the OS's IP and transport protocol implementation. For example, TCP timeouts might be configured to different default values causing mobile connections and tunnels to be held either shorter or longer. Also, mobile network firewalls have been found to interfere with transport and application layer timeout and keep-alive or heartbeat mechanisms on mobile devices [Wan+11b].

The actual user-traffic patterns are generated by the applications running atop the OS. An example for how applications can influence network signaling is the aforementioned "Angry Birds" with its ad-retrieval strategy causing network traffic and possibly signaling in certain intervals. Since the application ecosystem for smartphones is extremely rich and ever growing we cannot pinpoint individual ones from our aggregate dataset.

An additional factor in the picture is the user and her or his behavioral patterns. They express themselves both in the traffic dynamics and in the mobility pattern, but they are rather difficult to distinguish in such a dataset given the large amount of data and

the difficulty of correctly correlating tunnel management messages. We leave this as potential future work.

Easier to observe are the temporal effects of user behavior, which do not target individual users but the overall effects of a device's usage based on the time of day, the day of the week, or other time spans. In network user traffic analyses diurnal effects are typically very distinct with peak traffic some time during the day and the lowest traffic shortly after midnight. But these investigations are for user traffic only. We aim to find out, if the mobile network control plane shows similar patterns and can thusly be correlated to user traffic.

We also expect the mobile network and its protocol implementations to express themselves in the measurements. For example, the RRC idle timer is typically in the range of 10 to 30 minutes, which could mean there will be a large number of tunnels with a duration in this range. Such choices are usually made either by the mobile network operator or the device manufacturer and can vary from one implementation to another. It is therefore quite difficult to give any hard numbers in advance, and one has to correlate such aspects with certain events in the results.

3.5 STATISTICAL FOUNDATION

- The Art of Computer Programming Volume 2 (3rd ed.) random numbers and statistical tests[Knu97]
- Probability Distributions
- Null / alternative hypotheses
- Statistical model
- probability distribution fitting methods: moment matching, maximum likelihood
- tests: sum of squares, variance, chi-squared, kolmogorov-smirnov
- visual tools: histogram, density, ecdf

3.5.1 *Distribution Function*

$$F(x) = P(X \leqslant x) = \text{probabilty that } (X \leqslant x). \tag{3.1}$$

Properties: monotonous
empirical distribution function $F_n(x)$ for values $X_1, X_2, ..., X_n$

$$F_n(x) = \frac{\text{number of } X_1, X_2, ..., X_n \leqslant x}{x} \tag{3.2}$$

3.5.2 *Distribution and Function Fitting*

One of the analysis's goal is to break down the actual measured system to a simplified model. Usually this is conducted by finding matching random distributions using one of several readily available matching methods which rely either on closed formulas or numerical optimization.

MATCHING MOMENTS Parameters for a selected distribution are estimated by calculating the first and higher moments of the given random variables and solving equations corresponding to the selected distribution.

http://cran.r-project.org/web/packages/fitdistrplus/fitdistrplus.pdf Vose D (2000) Risk analysis, a quantitative guide. John Wiley & Sons Ltd, Chischester, England, pp. 99-143.

MAXIMUM LIKELIHOOD A fit is found by calculating the log-likelihood of the given random variables for a selected distribution and maximizing the likelihood.

TODO: more details

EUREQA In cases where no "simple" distribution fit was plausible we attempted to match generic functions to the sample empirical distribution using tools specialized for this case.

TODO: optimization, quoting, short desc

3.5.3 *Statistical Tests*

Generally, tests compare the values observed in an experiment (in our case data obtained from measurements) to expected values following a theoretical distribution. In this case, the tests are used to validate and estimate the quality of the discovered fits to the empirical data.

3.5.3.1 χ^2 *Test*

Specifically this means Pearson's chi-square test for independence[Peaoo] and is the oldest known test. It can only be used for discrete count values obtained from independent observations and is compared against a frequency distribution. It is defined as

$$V = \sum_{i=1}^{k} \frac{(o_i - e_i)^2}{e_i}. \tag{3.3}$$

This simply calculates the sum of the squared difference between the observed o_i an expected values e_i and adjusts each for their weight. The result can then be compared to the χ^2-distribution with the same degrees of freedom[6] as the test for a given significance level. In most practical cases comparison is conducted against precomputed tables with set significance levels.

Most data collected in this thesis is typically continuous in nature on which this test cannot be used directly. However, data could still be split into a finite number of intervals, as is done when generating a histogram, and then using the intervals as categories for the chi-square test, albeit with a certain loss of precision.

3.5.3.2 *Kolmogorov-Smirnov Test*

This is where the Kolmogorov-Smirnov Test comes into play. First suggested by Kolmogorov in 1933 [Kol33] and expanded on by Smirnov in 1939 [Smi39] it is defined as

6 The degree of freedom of count experiments is one less than the number of observable categories.

$$
\begin{aligned}
K_n^+ &= \sqrt{n} \sup_{-\infty<x<+\infty} \left(F_n(x) - F(x)\right), \\
K_n^- &= \sqrt{n} \sup_{-\infty<x<+\infty} \left(F(x) - F_n(x)\right).
\end{aligned}
\tag{3.4}
$$

Once again the results are compared against a precomputed table of values from the Kolmogorov-Smirnov distribution to test the significance of the observed results' deviation from expected values. The advantage is being able to work directly with a measurement's continuous empirical distribution function.

Nowadays, there are also much more powerful statistical tests available. But, as will be explained later, these are not necessary (nor will suffice any better) for our kind of data.

3.5.4 *Sampling*

Taking randomly selected samples from measurement data does not only simplify handling large sets (working on a set with 2 billion entries is quite problematic) but can even improve statistical significance as long as one keeps in mind, that random sampling error can also be introduced using this. By selecting entries using a uniform distribution it is ensured that no unintentional sampling bias occurs. The intended evaluation is now applied onto multiple and independently drawn sample groups. If the results of every sample agree then it is also highly likely that the assumption holds for the whole data set.

TODO: R statistics book ref and information VERIFY

3.6 EVALUATIONS

In this section we attempt to shed some light on the overall control plane dynamics in a mobile core network. We evaluate a dataset recorded in a live 3G network for PDP Context durations, and attempt to show the possible impact of certain device categories on the total tunnel durations. As discussed before, this can serve as a proxy metric for the signaling load on the system.

3.6.1 *Factors Influencing Tunnel Durations*

With such a dataset available and with the intent to evaluate core network signaling by looking at tunnel durations, let's first discuss some of the factors that influence this duration.

One factor are the mobile devices themselves. The device decides when it should establish a mobile data connection, how long the connection is held, or which mobile technology takes preference. Devices can be further differentiated by their operating system and their firmware (sometimes called *baseband*) which usually takes care of much of layers 1 and 2.

Some specific tunnel durations could stem from the TCP/IP stack implementations in the operating systems of the devices. TCP timeouts might be configured to different default values in different releases of OSs. Also, mobile network firewalls have been found

to interfere with transport and application-layer timeout and keep-alive or heartbeat mechanisms on mobile devices [Wan+11b].

Of course, the applications that run on top of the OS and generate the actual user-traffic patterns play a role as well. An example for how applications can influence network signaling is the casual game "Angry Birds" mentioned before. Since the application ecosystem for smartphones is extremely rich (and grows still), we cannot pinpoint individual ones from our aggregate dataset.

An additional factor in the picture is the user and his or her behavioral patterns. They express themselves both in the traffic dynamics and in the mobility pattern, but they are rather difficult to distinguish in such a dataset given the large amount of data and the difficulty of correctly correlating tunnel management messages. We leave this as a potential future work.

We also expect the mobile network and its protocol implementations to express themselves in the measurements. For example, the RRC idle timer is typically in the range of 10 to 30 minutes, which could mean there will be a large number of tunnels with a duration in this range. Such choices are usually made either by the mobile network operator or the device manufacturer and can vary from one implementation to another. It is therefore quite difficult to give any hard numbers in advance, and one has to correlate such aspects with certain events in the results.

Based on these factors, it was decided to make a first categorization according to the device type, be it either a smartphone, a regular or feature phone, or one of the many 3G dongles or mobile routers. Second, we also differentiate based on the device operating system, if known. Both differentiating aspects should prove valuable for example in deciding if currently some phone types put more signaling load on the network and to direct measures to improve this situation. Pitfalls in this differentiation are described in the next sections.

3.6.2 *PDP Context Durations*

Our measure of choice are the PDP Context Durations as they carry lots of meaning in being directly related to the signaling amount in the network. Therefore, we now direct our attention at the tunnel durations in the individual device and OS categories as identified via TAC values.

3.6.2.1 *Tunnel Durations by Category*

Figure 3.9 shows the empirical cumulative distribution functions for the PDP Context durations in our dataset. We distinguish the total duration distribution as well as the the distributions for smartphones, regular phones, and 3G dongles. It can be observed that tunnel durations range between seconds and more than one week[7].

The median differs between device types, being much longer for 3G dongles than for mobile phones. This can probably be expected, as typical dongle sessions might involve working at a laptop for periods longer than a few seconds or minutes. Also for the dongles, we observe less extremely long tunnels with durations above several hours. Again, we could hypothetically relate this to a usual laptop working environment,

7 Although our dataset is one week long, some tunnels started before the beginning of that week, and ended within it. Since the tunnel start dates were still available from the system, we chose to include the data.

where the device is used for a few hours but then shut down. With this, the PDP Context is deleted as well. Interestingly, the median duration of regular phones is higher than that of smartphones. This may indicate that smartphones regularly (and perhaps automatically) cause data traffic and therefore tunnels to occur. We conjecture this to be a first indication of the "Angry Birds" effect of automatically transferring small amounts of data, e.g. weather reports, stock exchange data, RSS feeds, or email notifications. We also observe two distinct steps, one at 6.8 seconds for dongles, and one at 30 minutes in the overall and smartphone distributions. While we do not have a plausible explanation for the former, the latter could be explained by a value chosen for the RRC state machine transition to the IDLE state (cf. Figure 3.6).

Taking an even closer look at the smartphone device fraction, we can still observe major differences as depicted in the empirical cumulative distribution functions of Figure 3.10. The tunnel duration distribution of the Symbian device fraction behaves much closer to the regular phones already depicted in Fig. 3.9. A possible explanation could be the user-base being more traditional, or the devices being feature phones whose behavior clearly differs from smartphones.

Again, a number of steps are visible in the distributions. Those steps that are only visible in one operating system type point to a source involving the phone rather the network. This especially includes the 30 seconds, 300 seconds, and 600 seconds steps (i.e. accumulations of incidents) for Android, and the 600 seconds step for iOS devices. However, whether this behavior should be attributed to the operating systems themselves cannot be decided by only looking at these distribution. Other factors, e.g. the device's firmware version and user traffic dynamics need also be observed. We leave this point for future work..

A last artifact of note are the larger number of iOS devices with very short tunnel durations. Over 20% of all tunnels established by these devices are shorter than two seconds. While the actual cause still remains unknown, it could be an interaction between short regular traffic burst and 3GPP Fast Dormancy [GSM12] which iOS devices are known to implement. Fast Dormancy is a technique to release radio resources more quickly. It is deemed to improve device battery life, radio signaling and radio spectrum efficiency. However, due to the earlier and more frequent transition to the IDLE state, it also could cause an increase in core network tunnel management signaling, which is probably what happened in the iOS case depicted in the CDF.

3.6.2.2 *GTP Tunnel Duration*

In our evaluation, we define the duration of a GTP tunnel as the time between a GTP CREATE and the corresponding GTP DELETE event. After the reply for a CREATE has been sent from the GGSN any setup procedures at the node should have completed and it should recognize incoming traffic from or to this user. After the DELETE, the user's traffic will not be routed anymore. Any lazy cleanup happening after the DELETE is not relevant for this specific investigation.

We differentiate all the tunnel events in our dataset based on two factors. First, we look at tunnels from different device types, be it either a smartphone, a regular or feature phone, or one of the many 3G dongles or mobile routers. After that, we investigate possible influences from the operating system. Both categorizations should prove valu-

able for example in deciding if currently some phone types put more signaling load on the network and to direct measures to improve this situation.

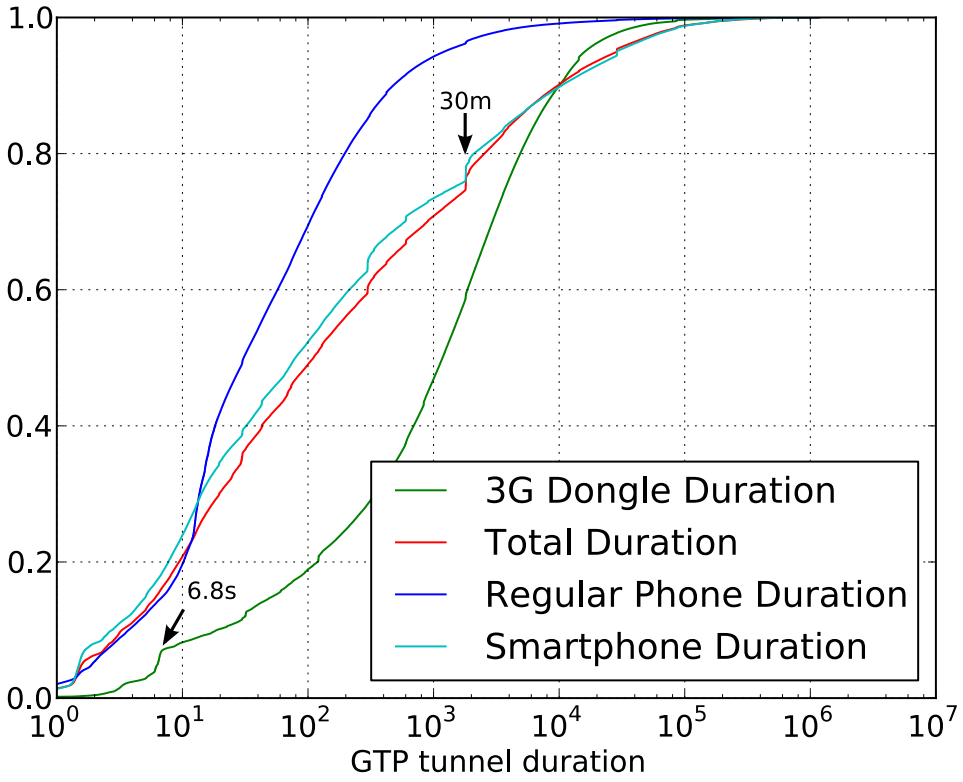


Figure 3.9: Tunnel duration distribution, separated for 3G dongles, smartphones and regular phones with medians at 115s (Total), 31s (Regular), 82s (Smartphone), and 1207s (3G Dongle)

INFLUENCE OF THE DEVICE TYPE Figure 3.9 shows the empirical cumulative distribution functions for the PDP Context durations in our dataset. We distinguish the total duration distribution as well as the the distributions for smartphones, regular phones, and 3G dongles. It can be observed that tunnel durations range between mere seconds and more than one week[8].

The median is clearly different between device types, being much longer for 3G dongles than for mobile phones. This can probably be expected, as typical dongle sessions might involve working at a laptop for periods longer than a few seconds or minutes. Also, for the dongles, we observe less extremely long tunnels. Again, we could relate this to a hypothetical laptop working environment, where the device is used for a few hours but then shut down. With this, the PDP Context is deleted as well.

Interestingly, the median duration of regular phones is higher than that of smartphones. This may indicate that smartphones regularly (and perhaps automatically) cause data traffic and therefore tunnels to occur. We conjecture this to be a first in-

8 Although our dataset is just one week long, some tunnels started before the beginning of that week, and ended within it. Since the tunnel start dates were still available from the system, we chose to include the data.

dication of the "Angry Birds" effect of automatically transferring small amounts of data, e.g. weather reports, stock exchange data, RSS feeds, or email notifications. We also observe two distinct steps, one at 6.8 seconds for dongles, and one at 30 minutes in the overall and smartphone distributions.

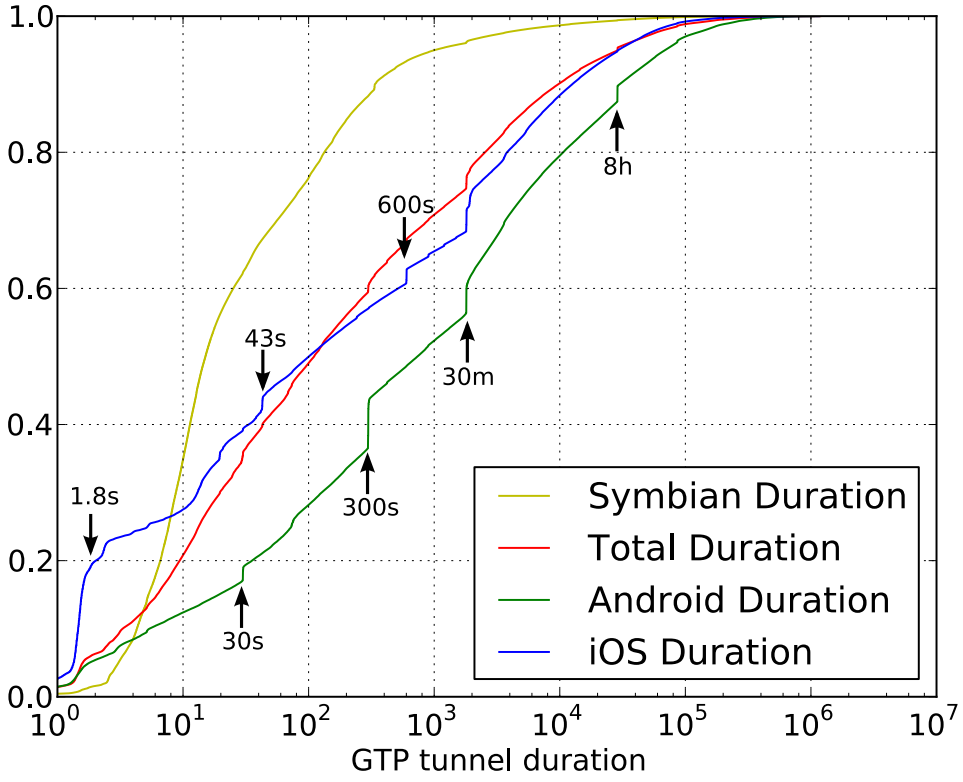


Figure 3.10: Tunnel duration cumulative distribution function, separated for Android and iOS devices; Medians at 115s (Total), 15.5s (Symbian), 104s (iOS), and 765s (Android)

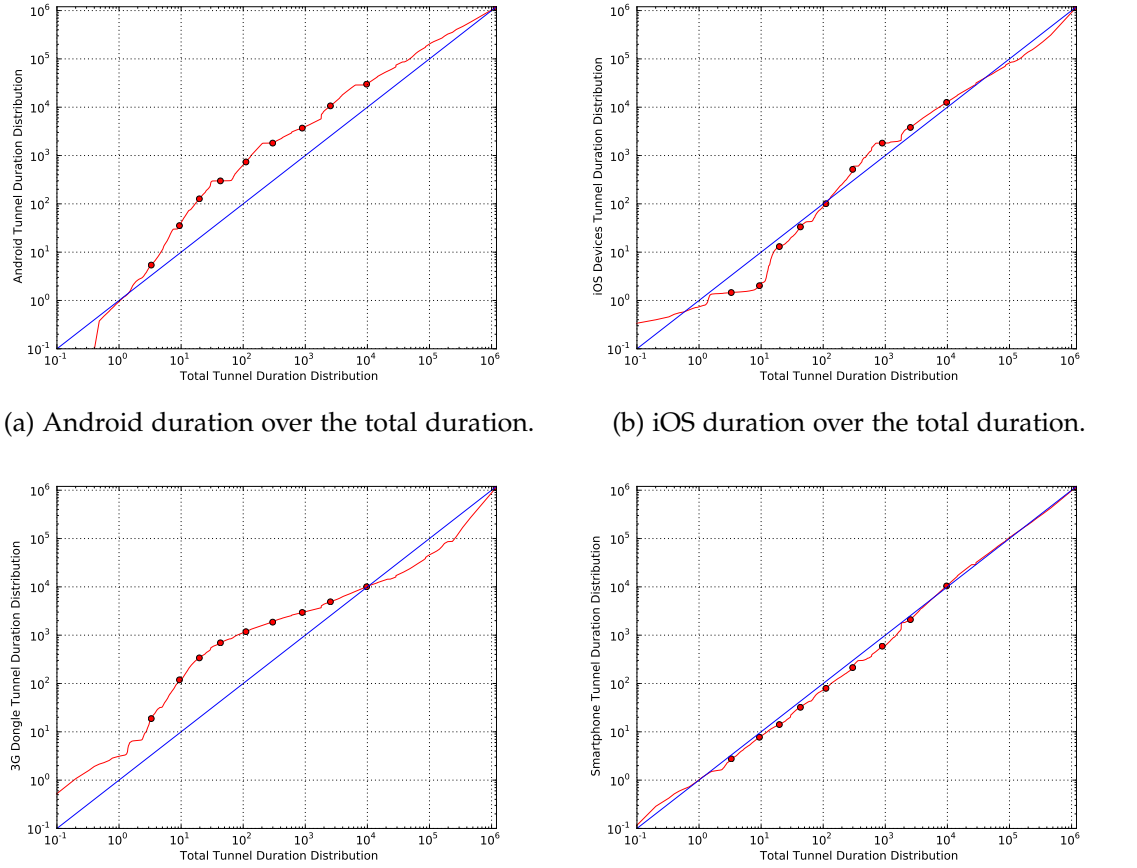
INFLUENCE OF THE OS Taking an even closer look at the smartphone device fraction, and differentiating the operating system to Symbian, Android, and iOS, we can still observe major differences as depicted in the empirical cumulative distribution functions of Figure 3.10. The tunnel duration distribution of the Symbian device fraction behaves much closer to the regular phones already depicted in Fig. 3.9. A possible explanation could be the user-base being more traditional, or the devices being feature phones whose behavior clearly differs from smartphones.

Again, a number of steps (i.e. accumulations of incidents) are visible in the distributions. Those that are only visible in one operating system type point to a source involving the device rather the network. This especially includes the 30 seconds, 300 seconds, and 600 seconds steps for Android, and the 600 seconds step for iOS devices. However, whether this behavior should be attributed to the operating systems themselves cannot be decided by only looking at these distribution. Other sources, e.g. the device's firmware version and user traffic dynamics need also be observed.

A last artifact of note are the large number of iOS devices with very short tunnel durations. Over 20% of all tunnels established by these devices are shorter than two

seconds. Our working hypothesis is, that this is an interaction between short regular traffic burst and a form of Fast Dormancy [GSM12] which iOS devices are known to implement, a technique to explicitly release radio resources. It is deemed to improve device battery life, radio signaling and radio spectrum efficiency. However, due to the earlier and more frequent radio state changes, it also could cause an increase in core network tunnel management signaling, which is probably what happened in the iOS case depicted in the CDF.

3.6.2.3 *Impact of Categories on Total Signaling*



(a) Android duration over the total duration.

(b) iOS duration over the total duration.

(c) 3G Dongle duratio over the total duration.

(d) Smartphone duration over the total duration.

Figure 3.11: Q-Q Plots of the tunnel duration distributions per operating system, with encircled deciles.

In an attempt to show which of the presented categories have an impact on the total duration (if at all), we present Q-Q plots of the various categorized durations against the total duration in Figure 3.11. In theory, if both durations follow the same distribution, one expects a straight line through the origin at an angle of 45°. A steeper incline indicates less densely spaced values in the distribution at the y axis. Looking at igures 3.11a and 3.11b which compare different operating systems, both similar and dispersing parts can be observed. While tunnel durations on Android are more similarly distributed for

the shorter and longer durations, iOS device tunnel durations are most similar to the overall tunnel duration distribution in the middle range of values.

Combining all types of smartphones together and comparing them to the other major player in any mobile network, the 3G dongles, we observe in Figure 3.11d that both the total and the smartphone durations are almost equally distributed (except for minor variations). On the other hand, 3G dongles follow a very different distribution, see Figure 3.11c. Their effect on tunnel management signaling seems to be negligible despite the large amount of traffic they are causing. Therefore, we conclude that planning and dimensioning of the control plane needs to keep smartphone behaviors more closely in mind than that of other device types.

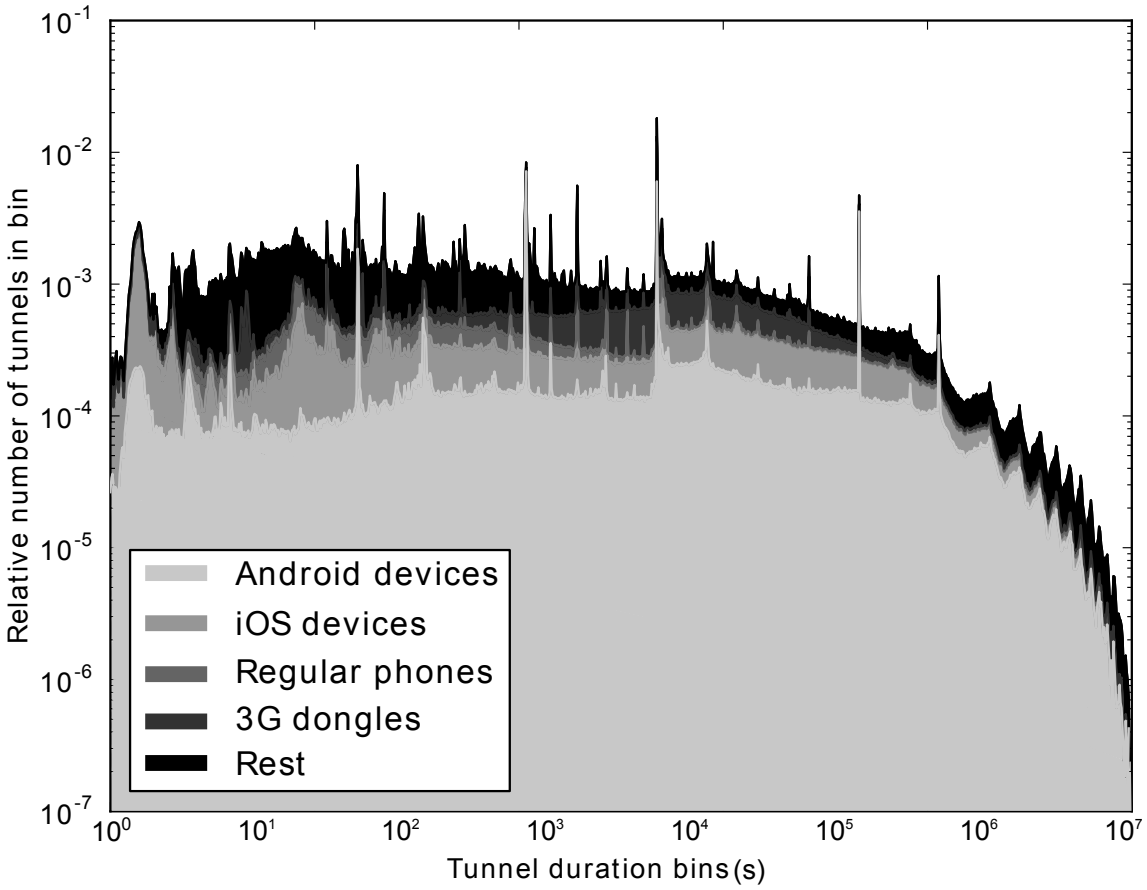


Figure 3.12: Stacked logscale bin plot of the number of tunnels with duration in this bin; classified by Android, iOS and 3G dongles.

Figure 3.12 shows another interesting influence the operating system has on signaling in the mobile core network. This plot shows the relative number of tunnels with a duration in one of 1000 logarithmically scaled bins, stacked by OS category on top of each other. As with the separate distributions, we discover that the durations are not evenly distributed, but rather follow sharp spikes. The largest spike across all categories is the one at a duration of 30 minutes, making up about 1.8% of all tunnels in the network. Since this spike happens across all device types, we think this makes a rather strong case for being network-induced, and an indication for the aforementioned possible IDLE state transition. On the other hand, the bulk in the short-to-medium ranges of tunnel duration is rather not governed by the two major smartphone operation systems

but by other devices in the network, which do not show major spikes in other bins. We can also recognize a long-tail behavior in the distribution of tunnel durations.

3.6.3 *Core Network Load Statistics*

Having characterized the dataset available to us we now shed some light on the control plane and load dynamics in a mobile core network and attempt to show the possible impact of certain devices or other properties of the network.

To examine some of these factors, we present the following number of individual investigations. Our measure of choice are the GTP tunnels as they carry lots of meaning in being directly related to the signaling amount in the network. We investigate their duration as well as the number of arrivals and look at a measure for the processing time of events at the GGSN. These insights will also allow us to build a simple toy model for the core network load in the next chapter.

3.6.3.1 *Tunnel Number of Arrivals and Inter-Arrival Time*

While tunnel durations and the involved signaling at the beginning and end of the duration is one aspect of control plane load, the number of tunnel arrivals might be another, which we are looking into in this section.

In addition to describing the arrival process on the basis of the number of arrivals, we also take a look at the tunnel inter-arrival time. Specifically, with this process we mean the arrival of tunnel requests, i.e. GTP CREATE requests, at the GGSN. This also adds to the foundation of the load model constructed in the next chapter.

Figure 3.13 depicts the number of arrivals per second during the whole weeklong period. Of note is the clear bimodal nature with one peak around twelve and the other in the low thirties. While the distribution is rather compact around these two peaks, there are some clear outliers up to 107. If we again hypothesize that an increased number of arrivals means higher load in the network, we can assume that load is not constant but rather switches between two modes with some periods with extraordinary load induced by an increased number of arrivals.

To find the cause of these two modes we take a peek at the diurnal arrival pattern. Figure 3.14 contains a violin plot showing again the arrivals per second but broken down by time of day. A violion plot, while being similar to a box plot, additionally shows the density of the individual items on the vertical axis. The nocturnal median from around midnight to 5am and the longer daytime median, 8am to 19pm, closely resemble the two modes found in the histogram. In between are short transition phases. Notably, during daytime the arrivals and their densities are spread out on a much larger value range. This could be an indication of load fluctuations in the system.

To investigate the arrivals from yet another angle we take a look at inter-arrival time of the tunnels in Figure 3.15. This metric is more suited to describe the arrival process in the toy queuing model we propose. The empirical CDFs are again broken down by time of day, the same diurnal load oscillation can be observed. The medians range between about 20 and 60 milliseconds. Figure 3.15a, which represents all tunnel requests that the GGSN received, shows wave-like steps in 20ms intervals in the plot. As this is happening very regularly at every time of the day we believe, that this effect must be

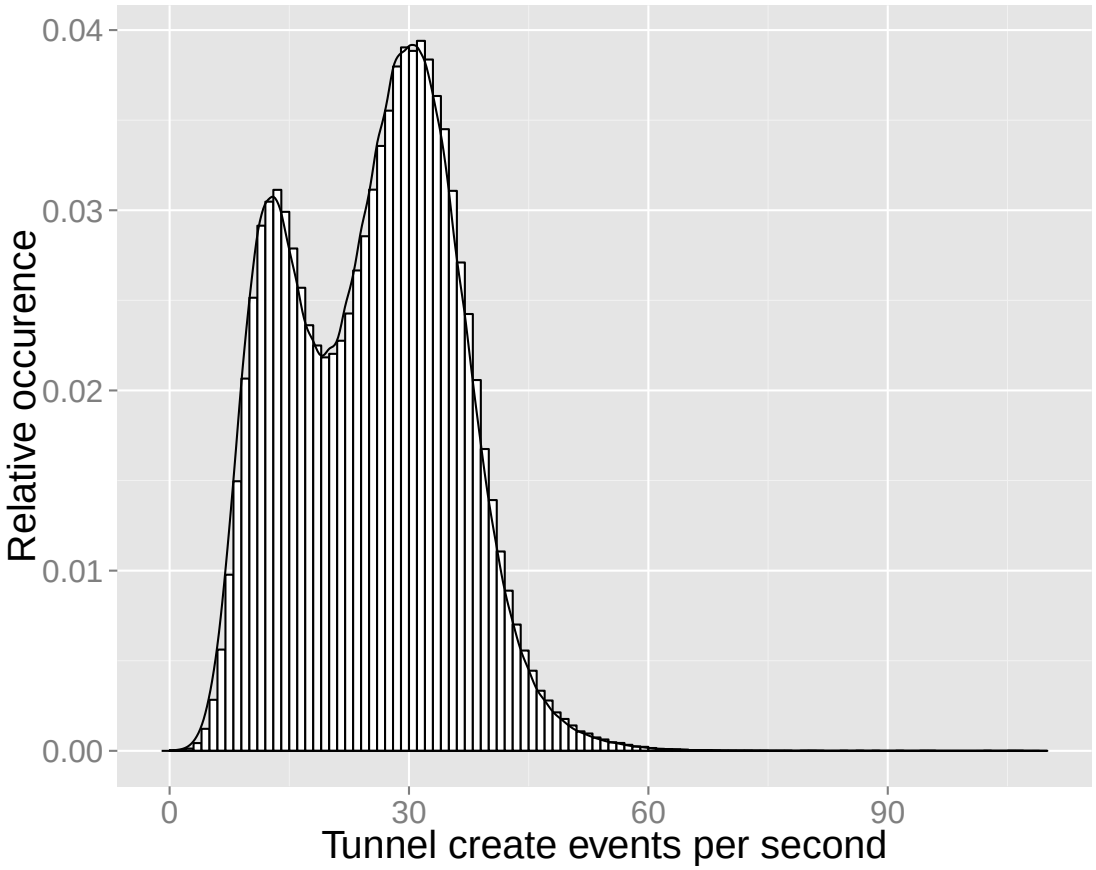


Figure 3.13: Tunnel arrivals in one second intervals.

from a source inside the mobile network and not induced from the outside, .e.g. through mobile devices.

This becomes even more peculiar when further breaking down the tunnel arrivals. We now distinguish between active tunnels, i.e. tunnels, that actually transported user traffic during their lifetime (cf. Fig. 3.15b), and active tunnels, which were created while having a GPRS (Fig. 3.15c) or UMTS (Fig. 3.15d) connectivity, respectively. Note that only about 86% of requested and created tunnels where actually used for user data transmissions afterwards. The 20ms-steps occur strongest when observing all tunnel arrivals, in a weaker form it is also present in the active and UMTS tunnel portion.

Our working hypothesis as to the origin of the effect is the Transmission Time Interval (TTI). This time indicates the duration of a radio transmission and is usually either 10 or 20 milliseconds in length. It is also in sync for the whole network of base stations making the TTI noticeable even when not measuring directly at the radio link. The observed step-width of 20ms therefore indicates, that the signaling procedure the GTP CREATE is part of, includes at least one trip from the mobile device over the radio interface. This makes sense, as the tunnel is typically created during the GPRS Attach procedure, which is indeed initiated at the user's device. Unfortunately, this also makes the tunnel arrivals come in somewhat batched, which could momentarily increase the load at the GGSN that then would need to process more requests at once than if the arrivals followed a smooth stochastic distribution.

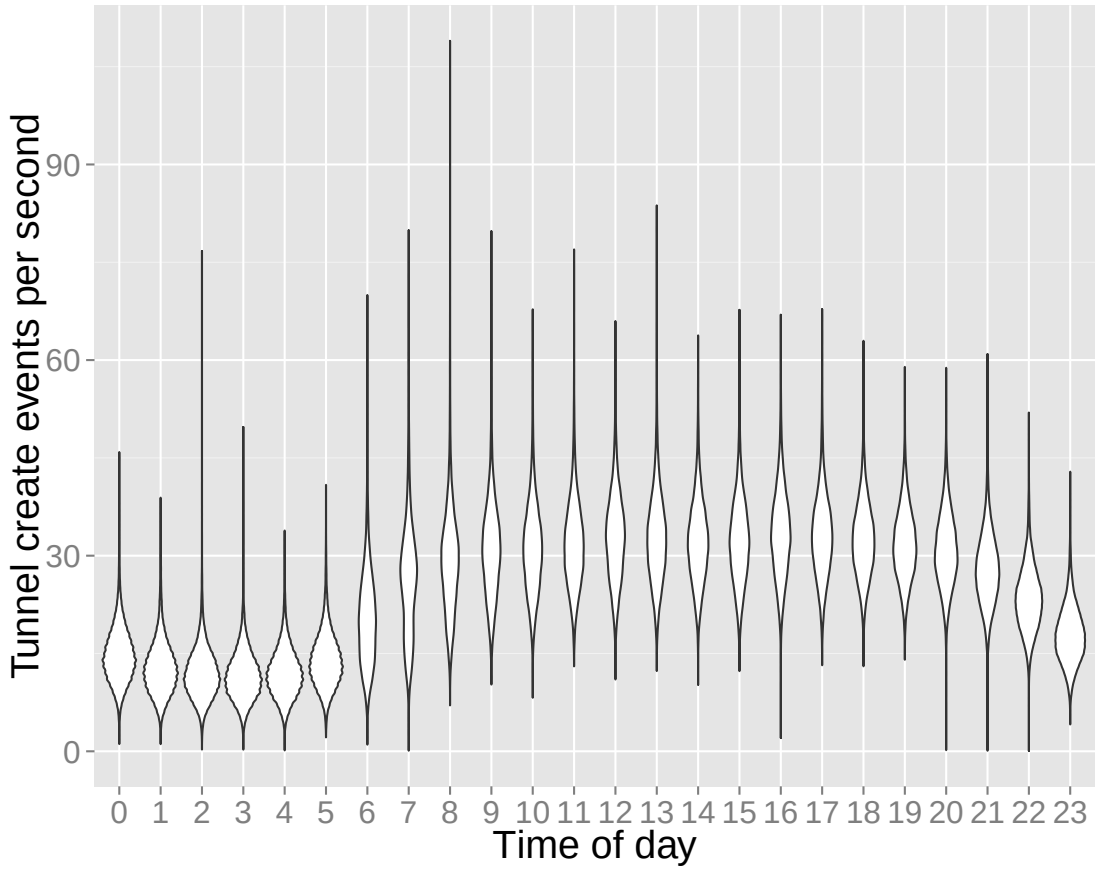


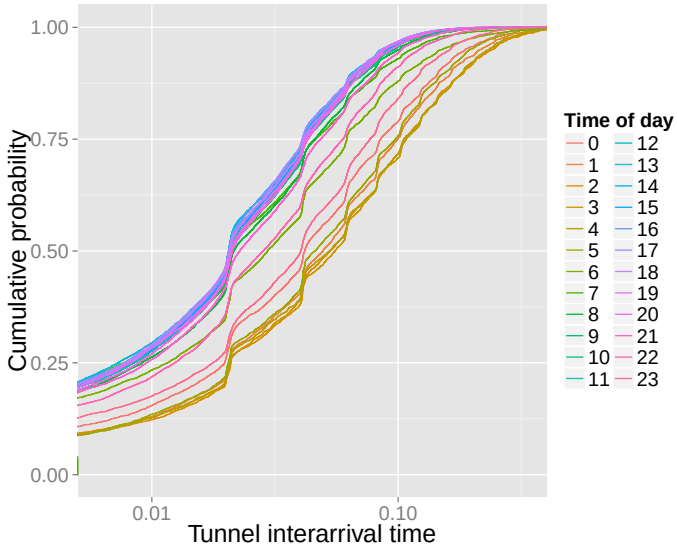
Figure 3.14: Violin plot of tunnel arrivals in one second per time of day.

3.6.3.2 *Tunnel Event Processing Time*

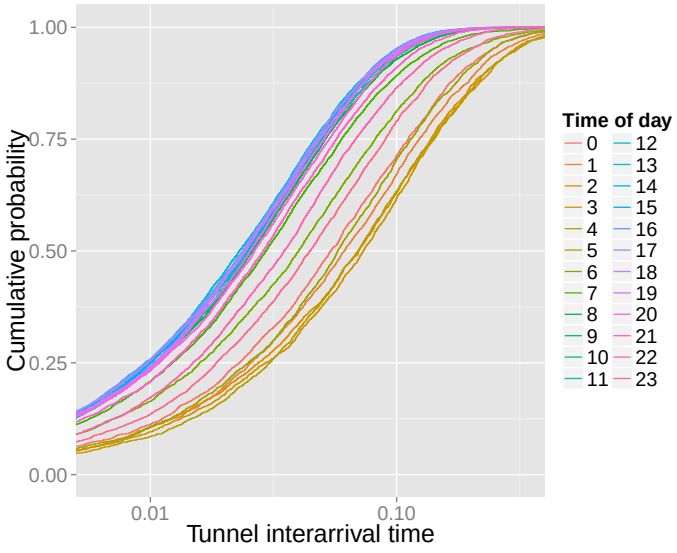
This brings us to another and potentially more direct measure of GGSN load, namely the event processing time, meaning the time it takes for the GGSN to fulfill a GTP request. This is calculated from the requested and finished timestamps of every GTP event in our dataset. As the measurement is conducted at the Gn interface these timestamps represent the time the GTP signaling request moves to the GGSN and the time the response transitions through the link.

As stated in the previous section, it would be of special interest to know if the setup time of tunnels is influenced by anything, as this is one of the GGSN's most time-sensitive jobs and can impact the time a user has to wait before being able to actually transfer data. Unfortunately, some issues with the dataset did not allow the investigation of the processing time of either create and delete messages.

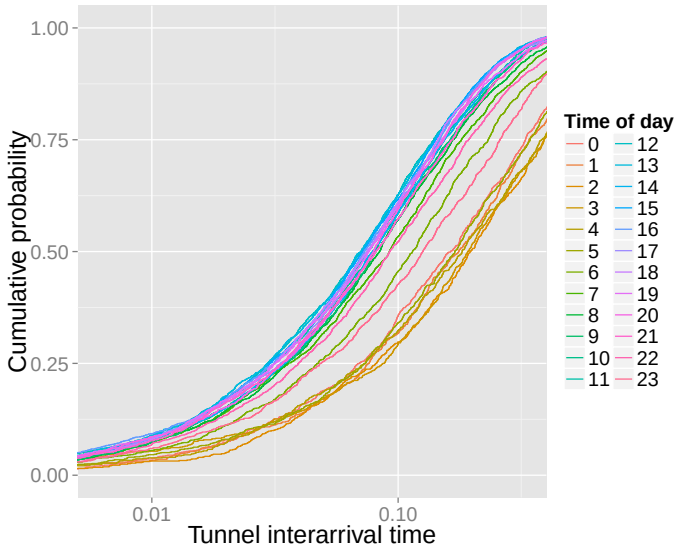
However, we could investigate the processing time of GTP update messages. The core network transmits roughly two orders of magnitude more update than either create or delete events and therefore the number of usable events exceeded the significance level. While no direct investigation of the setup and deletion procedures was possible with these events, a rough overall picture of load can still be attained through this. Figure 3.16 depicts a band of empirical cumulative distribution functions for the processing time of update events broken down by time of day. The processing time is almost uniformly distributed between 2 and 22 milliseconds, with a slightly longer duration during the



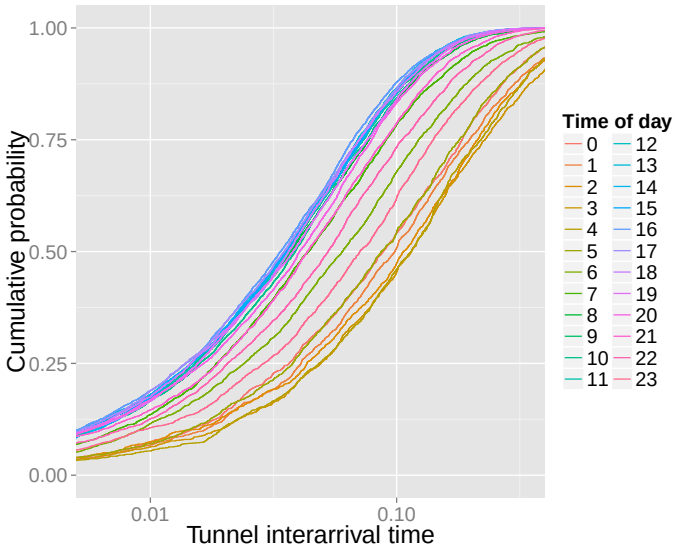
(a) All tunnel requests.



(b) Only tunnels with data flows.



(c) Tunnels with data flows initiated in GPRS.



(d) Tunnels with data flows initiated in UMTS.

Figure 3.15: Empirical cumulative distribution function of the tunnel inter-arrival time in seconds by time of day for each day of one week.

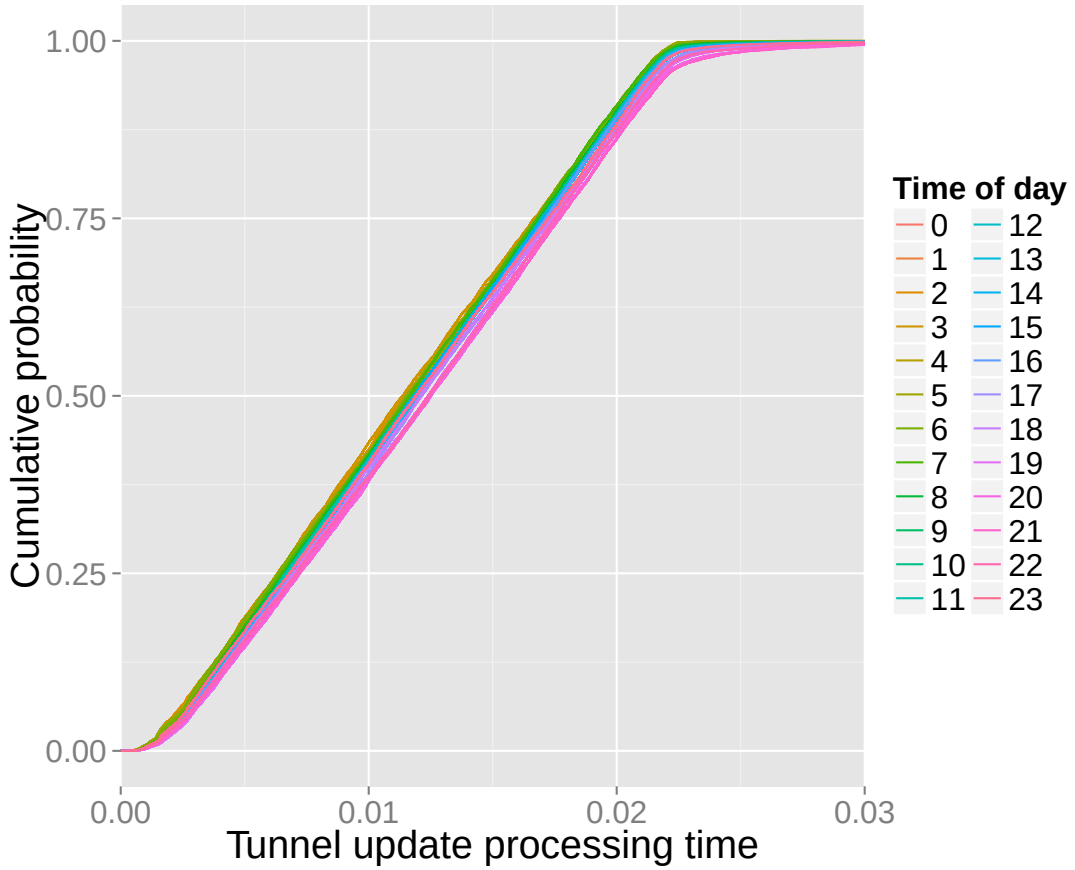


Figure 3.16: Empirical CDFs of the time it takes a GGSN to process a GTP update event, plotted for each hour of the day.

evening, making for a continuous uniform distribution. This is rather unexpected as uniform distributions do not usually occur in computing processes. According to the central limit theorem one would rather expect to see a normal distribution influenced by, e.g., scheduling or queuing artifacts. In the future we hope to investigate these features more closely, including a proper investigation of the tunnel setup and teardown processing time, if the dataset allows it.

3.6.4 *Statistical Evaluation and Data Fitting*

Using this dataset, we can obtain the distributions required for the models. At first, we take a look at the tunnel interarrival time in Figure 3.17. Typically, a device will only hold one tunnel at a time, but this one tunnel can be initiated and shut down in rapid succession, thus causing the aforementioned issues in the radio network. The arrivals also show a strong diurnal effect, closely resembling patterns present in the actual user traffic: A decline of arrivals, i.e. longer interarrivals, late in the night and during the early morning hours with a peak rate in the afternoon and early evening. To represent this time-of-day dependence in the model, the measurement was split into the four time slots displayed in the figure. Each slot was then fitted with an exponential distribution by way of moments matching. This results in the cumulative distribution

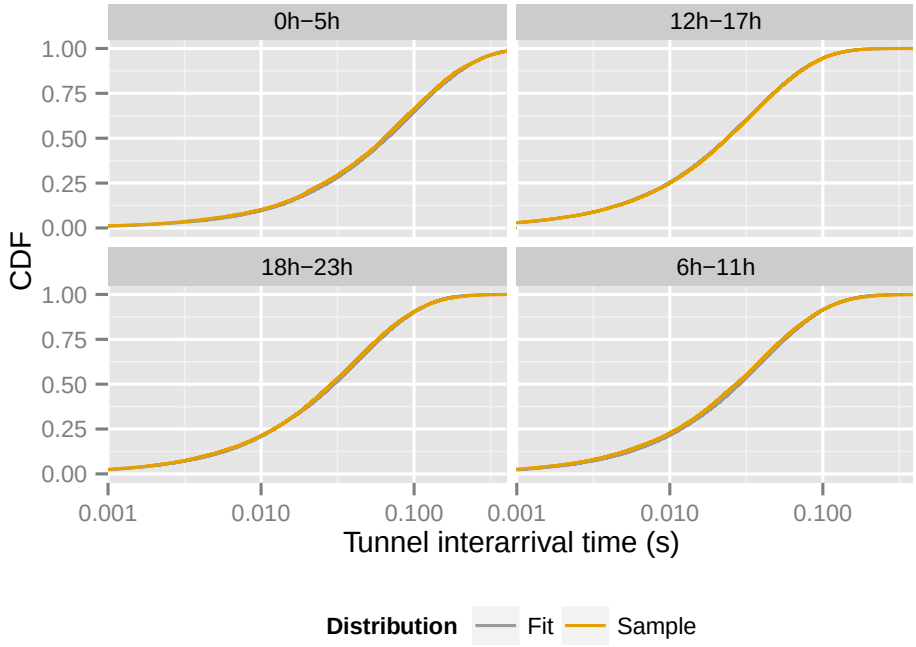


Figure 3.17: Empirical and exponentially fitted CDFs of the tunnel interarrival duration by time of day. CDFs are overlapping as the coefficient of determination is close to 1.

function $F(x) = 1 - e^{-\lambda x}, x \geqslant 0$ with λ given in Table 3.5 for the four time slots. The fitted functions match the empirical data quite well, with some deviation present at the left tail but overall with a positive correlation coefficient approaching 1.

The second important tunnel property is the duration the PDP Context state accompanying a GTP tunnel is held at the GGSN. Fig. 3.18 shows the tunnel durations split up for the time of day, as there is once again a slight diurnal effect present, albeit with shifted peaks. Longer tunnels tend to occur at night, shorter tunnels during midday. For the model, a distribution fit of the tunnel duration was also desired. However, none of the basic probability distributions (including exponential, gamma, and Weibull distributions) fit the tunnel duration well enough. One of the reasons for this probably being the correlation of the tunnel duration to a large number of factors, including user behavior and network-specific timers and procedures, introducing artifacts which make it hard to fit any distribution against. Instead, we fitted rational functions to the empirical CDF using Eureqa [SL09; SL13]. This allowed for a much closer fitting while still smoothing out some of the artifacts. Table 3.5 also displays these functions fitted to the inverse CDF, to be directly used for generating random numbers using the inversion method. Both the CDF in Fig. 3.18 as well as the Pearson correlation coefficient confirm the goodness of the fitted functions.

3.7 QUEUING THEORY BASICS

- Kleinrock Queuing Systems Volume 1 [Kle75]
- Tran-Gia Analytische Leistungsbewertung verteilter Systeme [Tra96]
- Markov Models

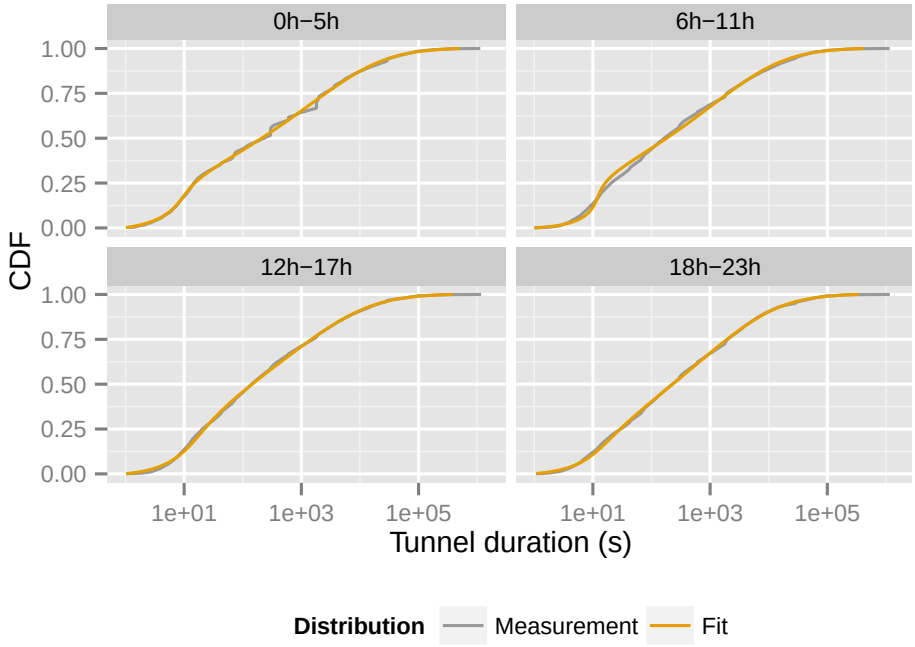


Figure 3.18: Empirical and fitted CDFs of the tunnel duration by time of day with fitted rational functions.

- Solvability and Queuing Simulation

3.7.1 *Little's Law*

"A proof for the queuing formula: L= λW" [Lit61]

With L as the number of customers in a stable system, the arrival rate of new customers λ and the average time W of a customer in the system this universal law states:

$$L = \lambda W \tag{3.5}$$

3.7.2 *Kendall's Notation*

Kendall's notation is a naming and classification convention for queuing systems first defined by Kendall in in 1953 [Ken53] and later extended on. In its simplest form it reads *A/S/s* with A denoting the arrival distribution, S the service time, and s the number of servers. One extended notation *A/S/s-q*, the one we will use, additionally describes the queue length. With this, one can, e.g., easily distinguish between a queueing system ($q = \infty$) and a blocking or loss system ($q = 0$). The most commonly used arrival processes and servie time distributions are summarized in Table 3.6.

The simplest queuing system is *M/M/1-∞*, which can also be described as a Markov chain and thus

TODO: oder direkt M/M/n-∞?

State probability, i.e. number of customers in the system Blocking probability p_B (for loss systems)

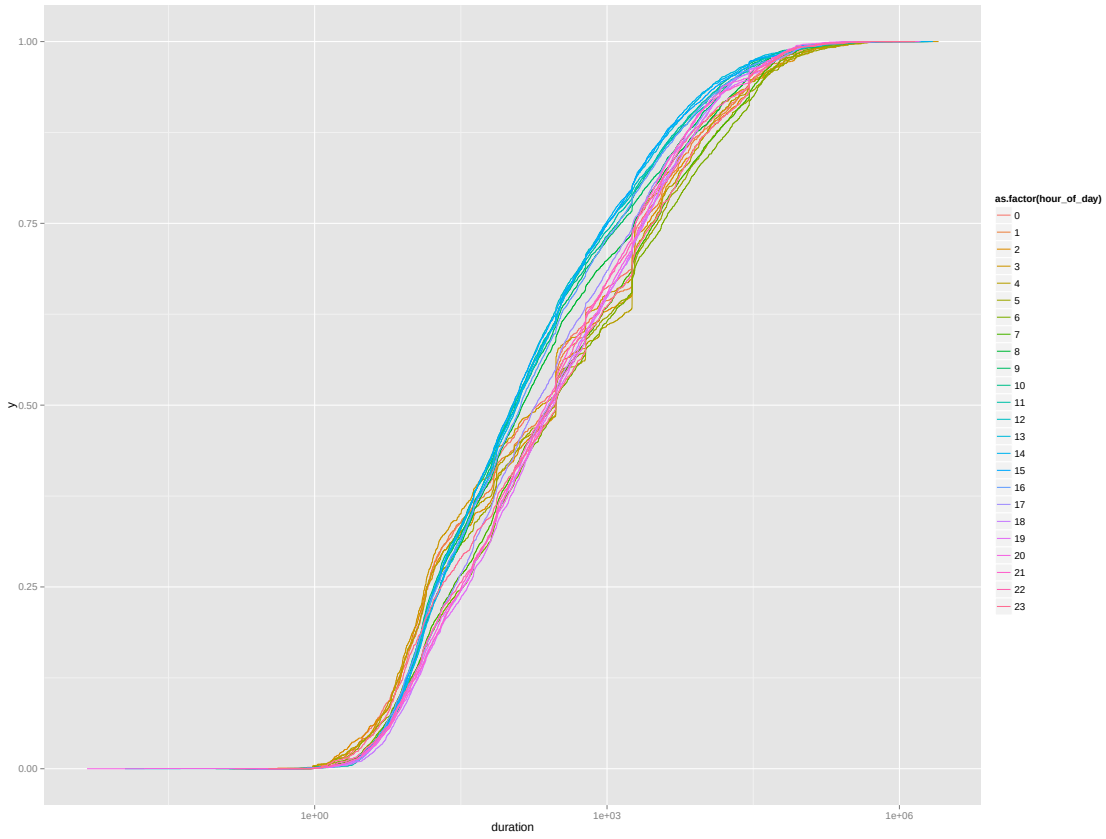


Figure 3.19: Tunnel duration of all active tunnels by time of day.

Following from Little's Theorem, the queue utilization ρ is given as

$$\rho = \frac{\lambda}{\gamma} \tag{3.6}$$

with λ Poisson arrival rate, γ exponential service time parameter
explain Erlang loss system and tractability

3.8 MODELING MOBILE NETWORK LOAD

Drawing conclusions from statistical analysis alone is a difficult task. The next logical step lies therefore in the creation of models abstracting this real system, making them easier to calculate with the loss of some precision. This and future improved models should support network operators in predicting the signaling load in their core network with the benefit of improved network engineering and correctly scaling core components.

With the increased importance of smart phones, mobile networks are currently experiencing rapid growth. Compared to a fixed access provider additional aspects have to be taken into account when dimensioning a mobile network. First and most prominent is the planning of radio access cells – their coverage, frequency selection, and backhaul, i.e. connection to the operator's network. Aside from substantial administrative and financial efforts this problem has been largely solved, radio network planning tools and research readily exists [Tut98]. Albeit of equal importance, there is much less public

knowledge and research on the second aspect in setting up the mobile network: dimensioning the core network. Consisting of a large number of specialized network nodes not available as of-the-shelf commodity hardware and in need of careful tuning to each other, correctly putting together the core is no small feat. Unlike fixed access, mobile access networks require much more state to be held, with the nodes having to signal any state-change throughout the network.

One major metric to consider in the dimensioning is the number of supported tunnels, i.e. connections to the Internet, of the GGSN. The performance requirements of the GGSN depend on factors like customers to serve, applications in the network, user behavior and devices used. These factors are, during dimensioning, either unknown or subject to change as user behavior evolves. But these network components are sold as static middleboxes and cannot not be easily extended with of-the-shelf hardware in order to account for new requirements. The newly introduced concept of Network Function Virtualization (NFV) [NFV13] suggests to harness technologies from cloud computing in the network. This would allow network operators to scale out, i.e. using additional low performance machines, instead of scaling up, which requires them to replace existing hardware with more powerful components.

The contribution of this work is threefold. First, we introduce models for both a traditional GGSN as well as a virtual GGSN using NFV. Secondly, we provide distributions for GTP tunnel interarrival times and durations, readily to be used in other studies. Finally, we study performance trade-offs when using a virtual GGSN, discussing different options to consider when using a virtual GGSN.

3.8.1 *Creating a Simple Toy Queuing Model*

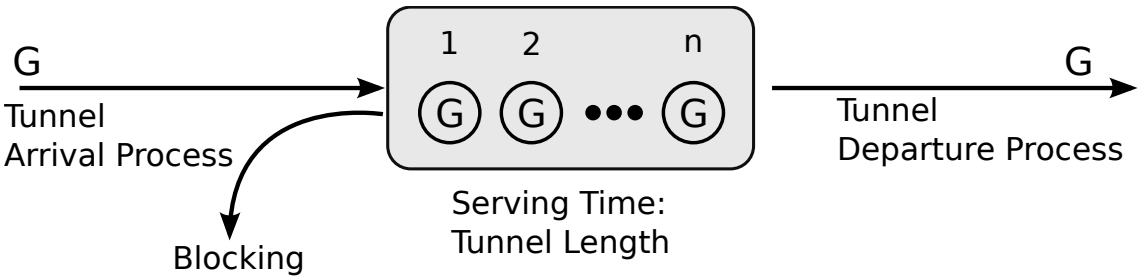


Figure 3.20: Simple toy-model for tunnel-induced load on the core network.

To begin the modeling process we attempt to represent the tunnel management as a queuing system, specifically as a G/G/n-0 system in Kendall's notation. Figure 3.20 shows this model for the case of our proposed tunnel load metric. Here, tunnels enter the system by a general random distribution, are then "served" at the GGSN for the duration of their existence, which also follows a general distribution, and leave the system, i.e. are torn down, afterwards. If the serving units are filled, blocking occurs and arriving tunnel requests are rejected.

In this case "servers" correspond to available resources at one or more GGSN, making the maximum number of tunnels hard to guess and depend on a number of factors. This could include soft-limits like the specific configuration, and hard-limits, e.g. the GGSN's processing and memory constraints. Unfortunately, all of these are unknown to us. Moreover, as the tunnels are all served on a relatively small number of hardware

entities they are not independent of each other. Increasing load could very well influence both the arrival as well as the serving process.

For the purpose of creating a toy model we are further simplifying the G/G/n-o to a M/M/∞ queue. As stated, no actual limit to the number of virtual servers is known and the data also does not show any obvious limits. So we can safely assume an unlimited system and do not have to treat blocking or queuing explicitly.

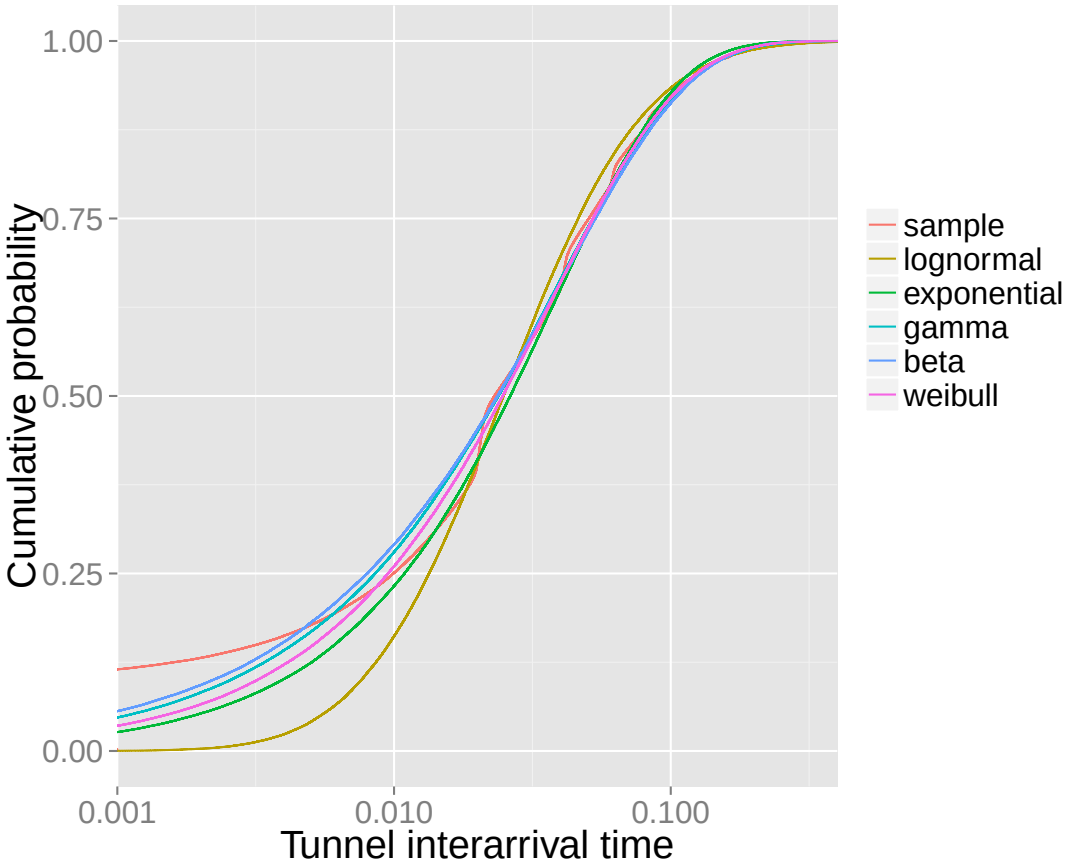


Figure 3.21: Sampled inter-arrival time CDF and fitted theoretical distributions.

Furthermore, we fitted univariate distributions to the experimental data for the tunnel inter-arrivals and durations and tested the goodness of the fit both numerically, using Pearson's χ^2 test, and visually for the density and CDF plots. No standard random distribution reaches the significance level for either process. We attribute this fact largely to the various artifacts in the data, e.g. the described wave effect every 20 milliseconds in the inter-arrival time. Matching them visually (confer also the cumulative distribution function plot in Figure 3.21) we find that the exponential fit is reasonably close to the experimental data in both the arrival and duration cases. Again, these distribution fits are just for a toy model to lay the groundwork for future and improved modeling.

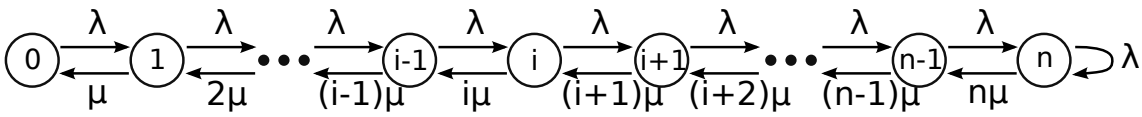


Figure 3.22: Markov chain model for the tunnel serving process.

Now, assuming both a Poisson arrival and an exponential serving process, a Markov chain representing the queue can be set up (cf. Fig. 3.22) and stationary analysis can be conducted. From the measured data an arrival rate of $\lambda = 25.64123$ and the parameter $\mu = 0.0001586728$ for the exponential service time distribution are calculated. Using Little's Law this gives an estimate for the mean number of concurrent tunnels at the GGSN of

$$L = \frac{\lambda}{\mu} \approx 161\,599.$$

As stated, the amount of state held at the node and propagated through the network is directly related to the number of tunnels. Therefore, we propose this metric as an initial estimate of the load at the GGSN.

3.8.2 *Advanced Models*

On the basis of this toy model better fitting models can now be constructed. Those should also factor in more of the core network's properties and specified parameters omitted in this model. Specifically, this means shifting from $M/M/\infty$ to the more generalized $G/G/n$ and therefore finding better distribution fits for the involved processes.

It is also entirely possible that the single queue approach is not the best way to describe control plane load. Several load influencing factors discussed earlier have direct influence on the tunnel arrivals and duration, e.g. the device type or the radio access technology. Therefore, amongst others multidimensional queuing networks or fluid flow could be a better fit. Our plan is to conduct further investigations into the modeling of mobile core network signaling. This also includes a rough simulative approach, which could also be used to validate our models against experimental data.

3.8.2.1 *Monolithic GGSN*

In this section we provide a model for a traditional GGSN and discuss a model for a virtual GGSN using NFV. In NFV [NFV13] static network middleboxes are replaced by commodity hardware. The tasks solved by the original middleboxes are then solved by dediciated software.

First, we give a model for a *traditional* GGSN, i.e. a network static network component. While we consider the GGSN to be one fixed entity, it can in reality consist of multiple servers. However, due to the fact that the GGSN is purchased from a vendor as a middlebox, idle servers can be neither deactivated nor reused for other purposes.

The queuing theory equivalent is displayed in Figure 3.23. New tunnels requests arrive according to a Poisson distribution with a rate of $\lambda(t)$ at the GGSN. This server will have a maximum tunnel capacity of c_c. When it is reached, blocking will occur and newly incoming tunnels are rejected. Traditionally, GGSNs can be expected to be overdimensioned in such a way, that this rarely happens. If the new tunnel is accepted, it will occupy one of the serving units of the unit for the duration $\mu(t)$ of the tunnel. As stated earlier, we can not model the tunnel duration to be markovian, resulting in a $M/G/c_c$ loss system. In order to give quality of service guarantees the network operator is interested in the system's blocking probability p_B, which we consider to be a key metric of our model. Additionally, the previously described diurnal patterns can

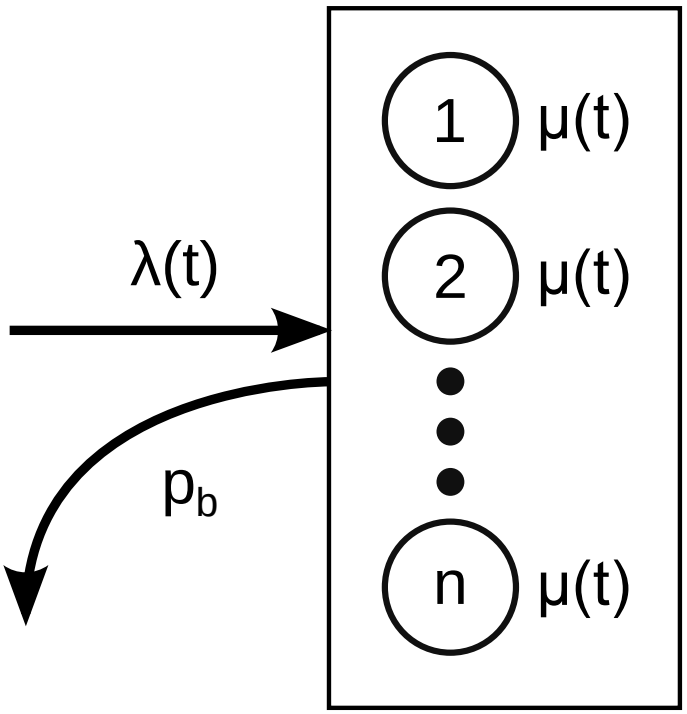


Figure 3.23: Model of a Traditional GGSN

are also be modeled by adjusting the arrival and serving process distributions for each time of day. This alternatively also allows just to investigate the busy hour and thus the system's peak load.

3.8.2.2 *GGSN using Network Function Virtualization*

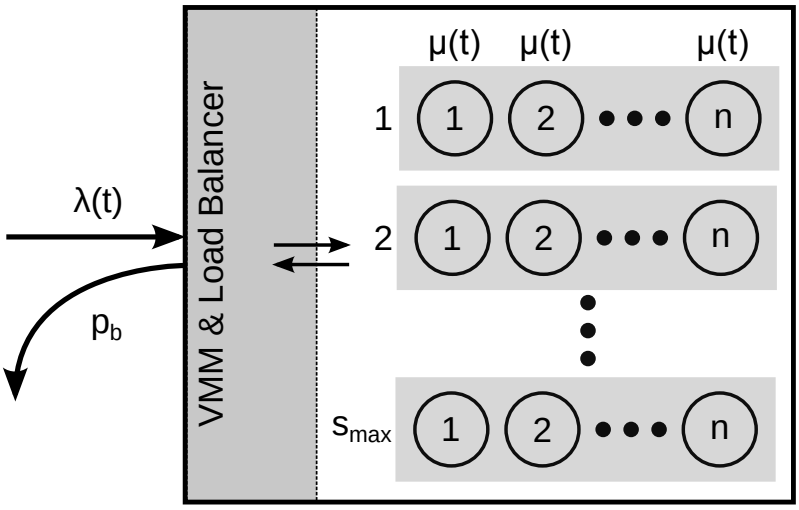


Figure 3.24: Model of a GGSN using Network Function Virtualization

In the second model, we introduce concepts from NFV, i.e. the idea to replace middleboxes with commodity hardware. This allows us to realize benefits from cloud computing, as we are now able to scale out, instead of up. The assumptions of the Markov arrival process $\lambda(t)$ and the serving time distributions $\mu(t)$ are carried over.

However, instead of one server processing every tunnel, this model assumes that there are up to s_{max} virtualized servers s_i. Each of these is much smaller than the traditional GGSN, having a tunnel serving capacity of $c_i \ll c_c$ and a total system capacity of $c_{max} = s_{max} \times i$.

In its initial state, for efficiency, all but a small portion of the server instances should be shut of. Only, when a certain condition is reached, a new one is provisioned. As a simple example, one could always hold one instance in reserve for upcoming requests and provision as soon as the reserver gets used. Similar rules should apply in the shutdown of servers and should form a hysteresis together with the boot condition. For example it would be possible to keep at least one server in reserve but never more than two.

If these conditions are not carefully selected and are in tune with the expected boot time of an instance, additional blocking can occur. Despite not having reached its maximum capacity, this system will still reject tunnel requests during the provisioning phase when no tunnel slots are free. This could be remedied by a request queue. However, this might just make the system more complex without providing real benefit, as mobile devices usually will repeat their attempts and would time out anyway when the request is taking too long.

To place incoming tunnel state on one of the available servers a load balancer is required. To ensure, that the system in run time can scale down to its actual needs, the balancer should place tunnels on servers, that are the fullest, keeping the reserve free. It may even migrate tunnel state from almost empty servers away so that these can be shut down, when the condition is fulfilled. Keeping instance close to their capacity should also have no impact on the performance a mobile device associated to a specific tunnel experiences. Adequate strategies for both load balancing and migration will be considered in future work.

3.8.3 *Simulative Validation*

3.8.3.1 *Testing the Model Numerically*

We implement the models using a DES with the SimPy [13] package as foundation. Our implementation is also publicly available[9] as a reference for future publications. To be in line with the measurement data we consider a simulation time of 7 days for all simulation scenarios, with a transient phase of 60 minutes accounted for. Ten replications of each scenario were performed. All error bars given in this section show the 5% and 95% quantiles of all replications.

We use the measurements in order to dimension a traditional GGSN as a baseline for all further studies. Based on these results, we examine the effects of network function virtualization by scaling *out* instead of up through a virtual GGSN model. Finally, we arrive at a more realistic version of the virtual GGSN by taking the start up and shut down times into account.

9 https://github.com/fmetzger/ggsn-simulation/

3.8.3.2 *Queuing Simulation Implementation*

3.8.3.3 *Traditional GGSN*

With the help of the interarrival times and duration of tunnels we study the traditional GGSN model previously introduced. Whilst our measurements provided us with information on the frequency of new tunnels and the duration they remain active, we have no reliable information on the number of active tunnels the GGSN can support. Thus, in a first step, we dimension the GGSN in such a way that a suitable blocking probability p_B can be achieved.

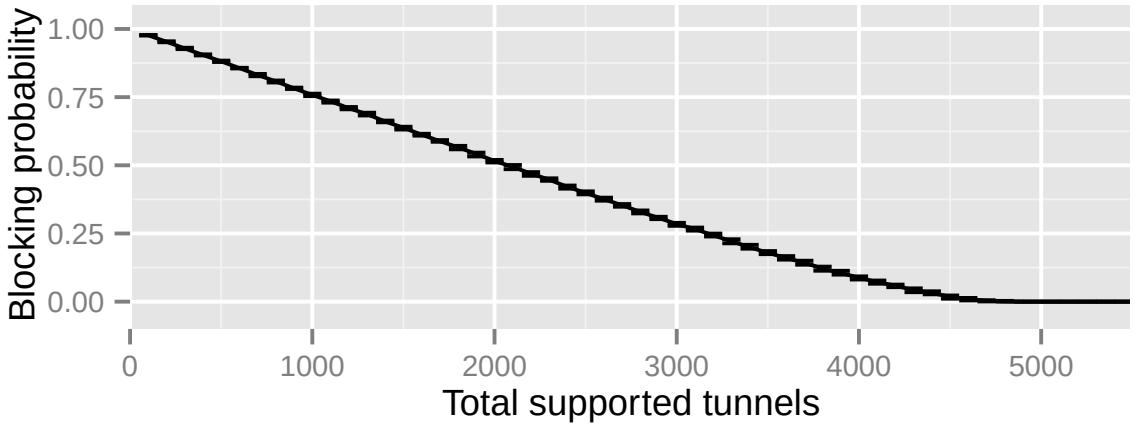


Figure 3.25: Impact of the number of supported parallel tunnels on the blocking probability for the traditional GGSN model. For each scenario the mean of all simulated replications as well as 5% and 95% quantiles as error bars are shown.

In Figure 3.25 the maximum number of tunnels n, that can be active simultaneously, is gradually increased to study the impact on the blocking probability p_B. We observe, that as the number of supported parallel tunnels increases, the blocking probability decreases. For the normalized interarrival no blocking is occurring if we allow for more than 5000 parallel tunnels. Thus, we consider the range of 4000 to 5000 parallel tunnels to be of special interest for the remainder of the study.

3.8.3.4 *Virtual GGSN*

In order to study the feasibility of the virtual GGSN approach discussed in Sec. 3.8.2.2, we compare the performance indicators of the virtual GGSN with that of a traditional GGSN. To this end, the virtual GGSN is simulated in varying configurations. The number of servers and supported tunnels per server is chosen in such a way that the results can be compared with those obtained from our study of the traditional GGSN. Due to simulation time constraints, only a representative subset of scenarios is simulated.

In the virtual GGSN model, servers are activated and deactivated on demand, while in the traditional GGSN model, the single server is always on. For this investigation a conservative start up and shut down time of 300 s is chosen. Generally, deactivating server instances reduces energy consumption and frees up inactive servers for other use. For this reason, the number of active servers is a relevant performance metric in the virtual GGSN model.

In order to analyze the influence of the different model parameters on the performance metrics, we perform a one-way ANOVA analysis with the results in Table 3.7.

High values for η_p^2 and Cohen's f^2 [Ell10] indicate that the main influence for both blocking probability and mean number of tunnels is the maximum number of tunnels n and servers S_{max}, i.e. the total number of possible concurrent tunnels in the system. Therefore, we study these parameters first.

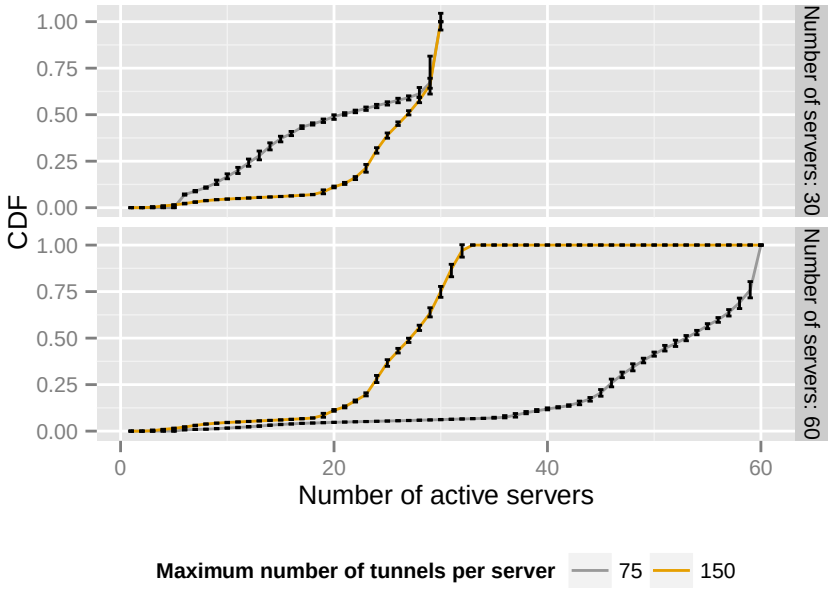


Figure 3.26: Impact of the maximum number of tunnels and number of servers on number of active servers in the virtual GGSN model.

In Figure 3.26 the Cumulative Distribution Function (CDF) of the number of active servers for four different virtual GGSN configurations is displayed. We observe, that increasing the number of supported tunnels per server allows a larger percentage of servers to be shutdown or used for other tasks. This demonstrates the scaling capability of the virtualized model quite well. Note, that both the scenario with 30 servers and 150 maximum tunnels per server as well as the scenario with 60 servers and 75 maximum tunnels per server share the same maximum amount of tunnels, 4500, being right at the center of the interesting range of candidates.

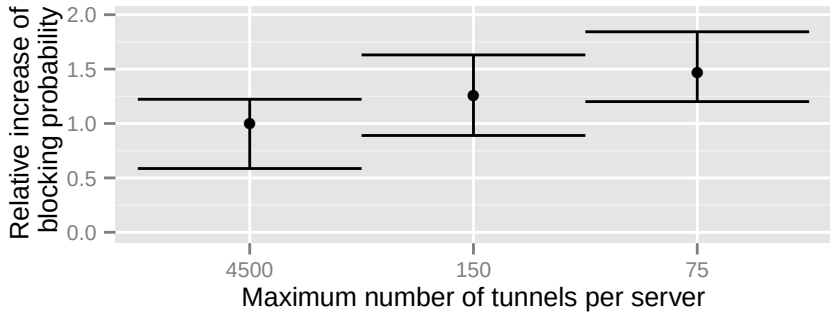


Figure 3.27: Relative increase of blocking probability on the number of servers compared to the traditional GGSN; with the 4500 maximum tunnels per server being on a single server, 150 on 30, and 75 on 60 servers.

Next, we take a look at the blocking probability of the virtual GGSN system in Figure 3.27 and compare it to the results from the traditional GGSN model. In Figure 3.27

we compare the blocking probability of the traditional GGSN system dimensioned for 4500 concurrent tunnels with the virtual GGSN.

We observe that, with the start up and shut down time of 5 minutes in mind, the blocking probability increases by a factor of 1.48 if the capacity of each server is set to 75, i.e. $\frac{1}{60}$ of the original server capacity, while 27 of all 60 servers can be turned of or used for other purposes at 50% of the time. We conclude, that choosing more powerful servers decreases the blocking probability but reduces the potential to disable servers.

So far we have considered a conservative start up and shut down time of servers of 5 minutes, which can potentially occur if current generation physical servers are used. In the next section we study the impact of reduced start up and shut down times with modern servers with fast storage (e.g. Solid State Disks (SSDs)) or virtual servers provisioned in the cloud.

3.8.3.5 *Impact of startup and shutdown times*

In this section, we first consider the impact of different boot and shut down times on resource utilization and blocking probabilities. We observe the impact of different start up and shut down times on both resource utilization and blocking probability. Afterwards, the influence of varying server start and stop times on a fixed combination of maximum tunnels and servers in the system is examined.

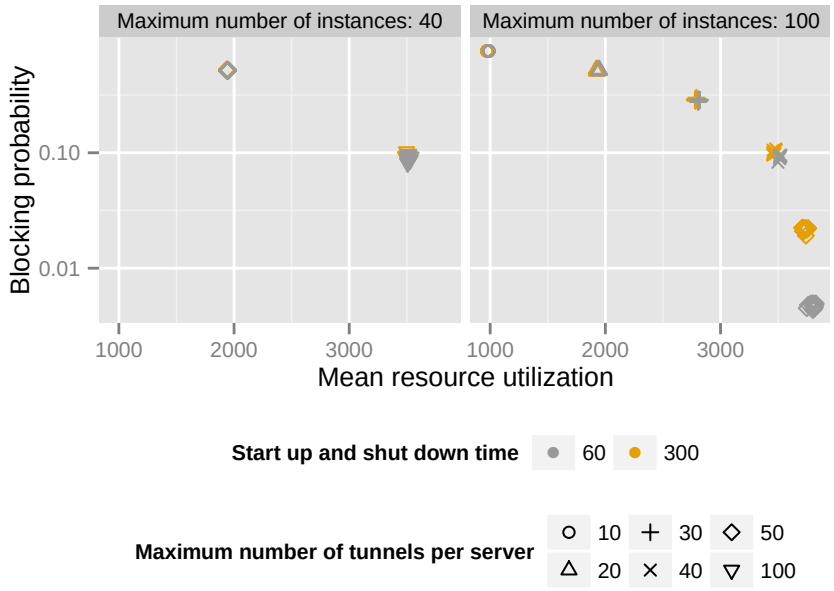


Figure 3.28: Trade-off between blocking probability and mean resource utilization with regard to maximum number of servers, maximum number of tunnels per server, and start up and shut down time.

Figure 3.28 shows scenarios with 40 and 100 number of virtual GGSN instances and 1000 to 5000 total concurrent tunnels. For each scenario, we study the impact of selecting different maximum numbers of tunnel per server as well as start up and shut down times on blocking probability and mean resource utilization. The first observation is that by increasing the number of servers, i.e. scaling out, the blocking probability can be decreased, while maintaining a relatively low mean resource utilization. In addition to

the previous effects, we notice that a higher start up and shut down time causes a slight increase in blocking probability for servers with low tunnel capacity.

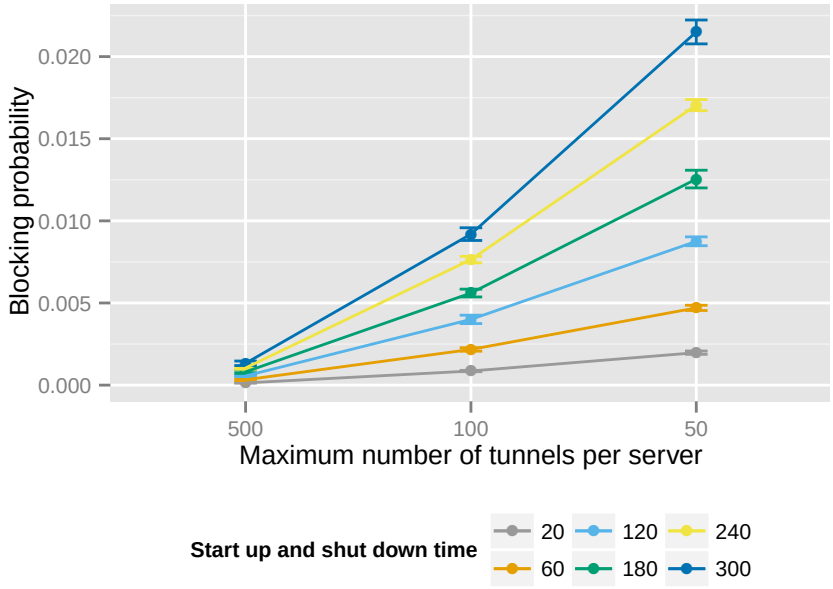


Figure 3.29: Influence of start up and shut down time on blocking probability with regard to different numbers of servers.

In order to study this behavior in more detail, we focus on a specific scenario in Figure 3.29, where 5000 total tunnels should be supported by the system. In order to achieve this goal, we consider three types of instances, with the server capacity varying between 50 and 500. In each case we change the start up and shut down time between 1 and 5 minutes. It can be easily observed, that lower server capacities combined with higher start up and shut down times increase the blocking probability. This is due to the server start up threshold mechanism, used in the model, not taking the additional capacity gained by activating an additional server into account. If a low capacity server with a long boot time is activated, there is a high probability that the system will quickly expend its capacity again.

Thus, it can be concluded, that if smaller instances are to be used, for example because they are cheaper than large instances, start up and shut down times should be kept minimal, for example by using virtual instances or SSDs.

3.8.4 *Modeling Discussion*

3.9 SUMMARY

In this paper, we took a look at the signaling behavior of devices in an operational 3G mobile network providing Internet access. Our focus does not lie on the wireless or user-oriented parts of the network, but on signaling in the core network. To the best of our knowledge, this paper is the first to offer an in-depth core network perspective on signaling. We give a GPRS and UMTS network primer, and introduce GTP tunnel management, explaining the causes and actions within the network. Our evaluation

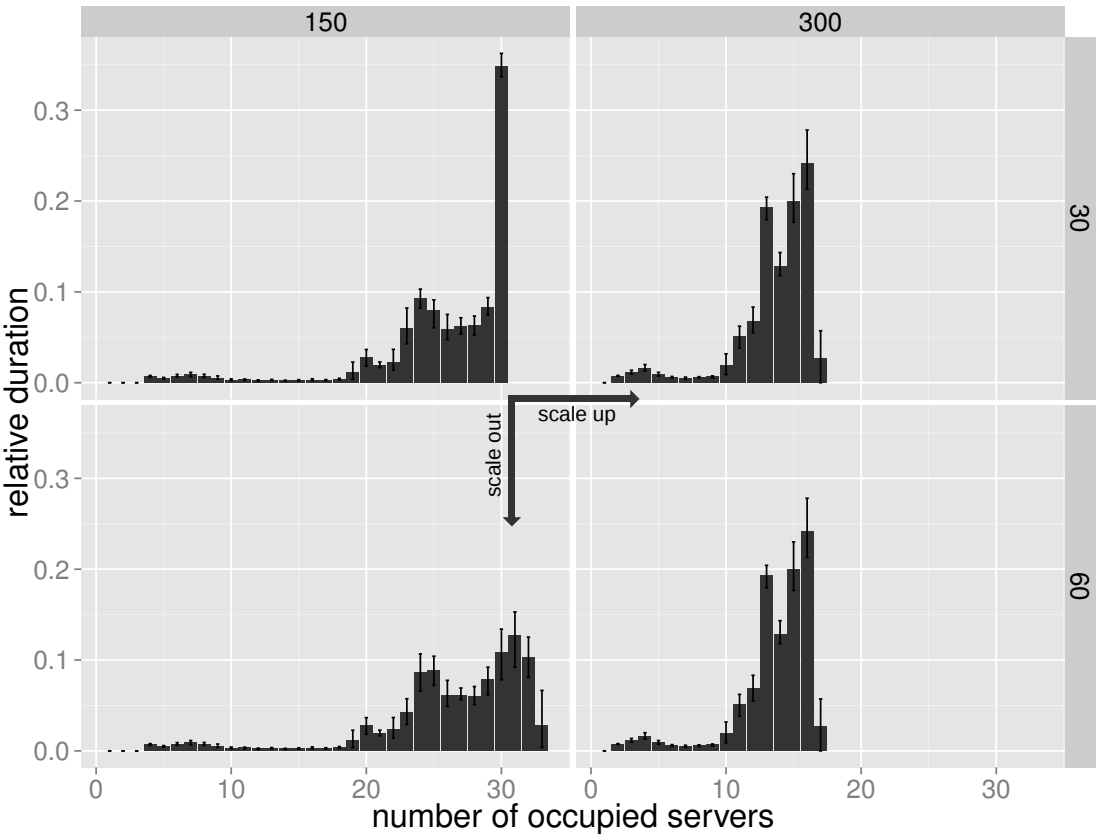


Figure 3.30: Resource usage from select maximum instances and tunnels combination, displaying the capability to scale.

is based on a week long data set acquired in the core network of an Austrian mobile operator.

In our observation of core network signaling involving PDP Contexts and their management, we looked at the effect of device types and operating systems on the duration of GTP tunnels. We can conclude that the distribution of tunnel durations in our evaluated dataset is dominated by smartphones. This is contrary to the conventional idea that a larger volume of user plane traffic also leads to an increase of signaling. In our dataset, this would mean that 3G dongles would cause most signaling, which is definitely not the case. Our paper shows that operators can determine which type of device has the most influence on the current network infrastructure by looking at and comparing tunnel duration distributions.

In this aspect, our findings support the stories of the casual game "Angry Birds" causing signaling storms in mobile networks by frequently downloading small ads, each small download resulting in disproportionate amounts of signaling load being generated. We conjecture from our results that measures taken to improve the radio interface control plane such as Fast Dormancy can have the converse effects in the core, as they could increase the tunnel churn.

All in all, our paper shows that operators can determine which type of device has the most influence on the current network infrastructure by looking at and comparing tunnel duration distributions. This investigations can also lead to better network plan-

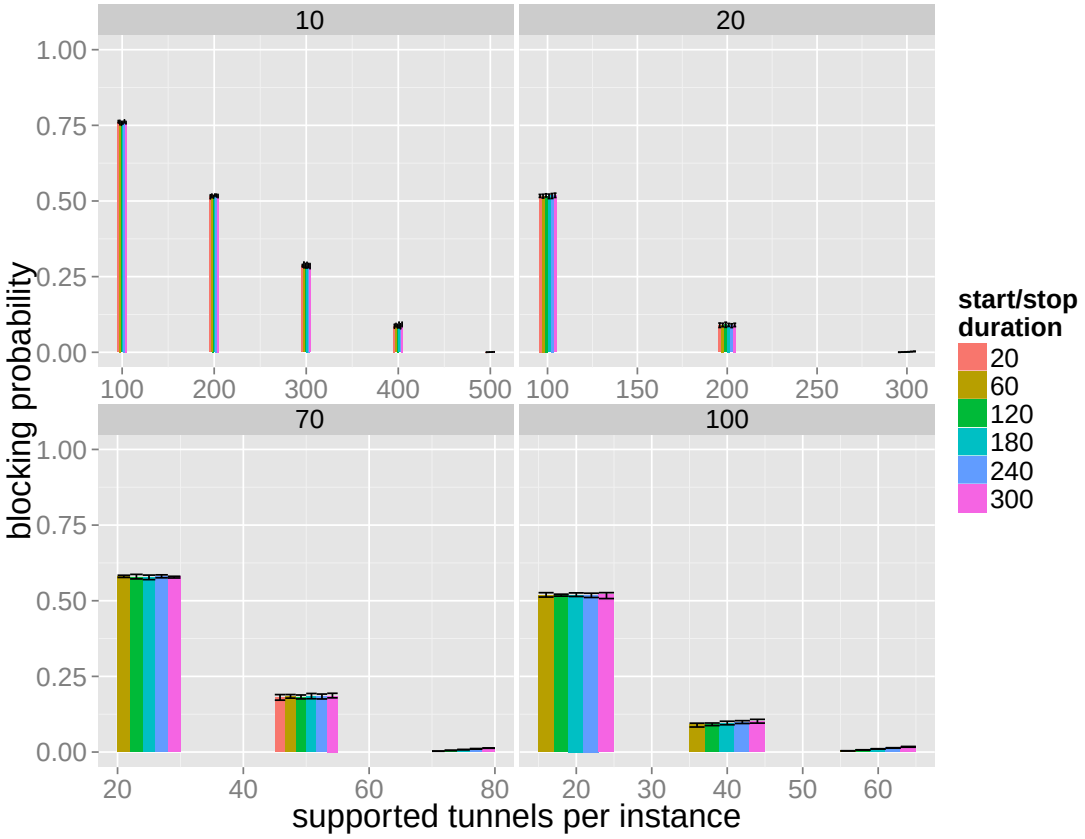


Figure 3.31: Influence of the boot and shutdown time on the blocking probability.

ning that is more aware of the control plane by providing the necessary tools to identify probable causes for control plane activity. Lastly, we hope to raises some awareness with programmers about the potential unintended side effects their application traffic patterns can cause.

This paper serves as an introduction to the topic of the 3G core network control plane, and therefore provides only some initial insights into the actual signaling dynamics. Therefore, we would like to expand our evaluations, as there are several angles not investigated so far that could prove worthwhile.

To get a grasp of the imposed load on the network as well as the involved network nodes, a calculation of the sizes of the tunneling messages was already hinted at. To improve on this naive attempt, actual numbers on the message sizes and involved IEs could be recorded in future traces. Having correct signaling traffic volume data still does not reveal the processing load on core network elements. We plan to improve our methodology in this respect by taking at a look at how long it takes for the gateway nodes to process GTP messages with respect to the current amount of user traffic and signaling. GTP tunnels also cause a certain amount of overhead through additional headers and potential fragmentation of the user traffic, providing another investigation venue for the future (albeit more oriented towards user-plane IP traffic).

Furthermore, besides the device-based classification, a differentiation based on the user traffic dynamics and correlation to signaling is planned. When looking closer at specific users, the mobility behavior also comes to mind. To investigate this, we intend

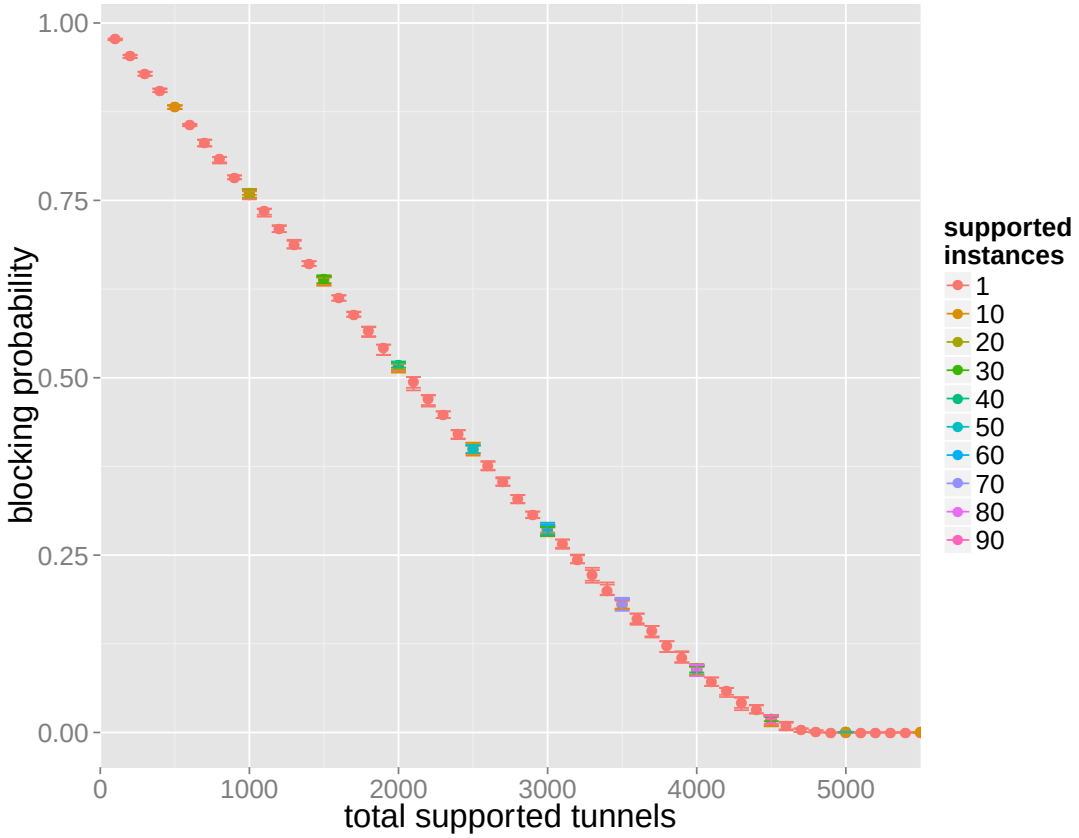


Figure 3.32: Comparison of the blocking probability of various server configurations.

to take a closer look at the occurring tunnel update messages as evidence, amongst others for mobility.

We also look forward to searching for multiple active tunnels per device. As discussed previously, the *Secondary PDP Context Activation Procedure* enables devices to establish up to ten additional tunnels attributed with a different, higher QoS level, if the network supports this. The additional load of managing and holding multiple tunnels plus the displacement of other, "lower-quality" traffic could prove to be an interesting investigation. Initial observations indicate that this feature is rarely used today by very few types of devices, but it will be of increased interest in the face of ongoing LTE/EPS deployments, whose specifications expand upon this secondary tunnel concept.

For additional load investigations we also looked at the inter-arrival and processing time of tunnels and found further evidence of radio and diurnal effects influencing the core network. With this data in mind, an initial $M/M/\infty$ queue was created to model load occurring at the GGSN with simple stationary analysis. This also serves a basis for future more detailed models.

We think that this investigation and load modeling can lead to better network planning: Being more aware of the control plane provides the necessary tools to identify probable causes for control plane activity. We would also like to expand our evaluations, as there are several angles not investigated so far that could prove worthwhile. This includes an examination of the exact number and size of signaling messages flowing through the core, a more detailed picture of the processing load these messages

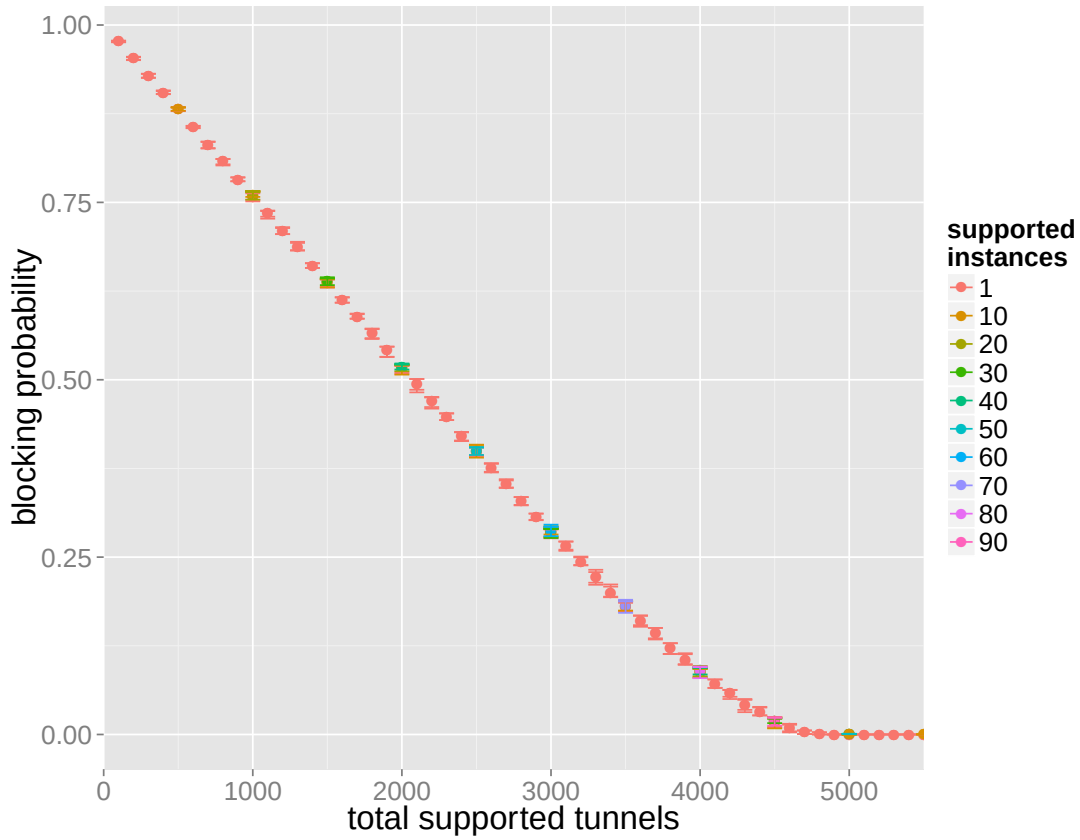


Figure 3.33: Comparison of the resource usage of various server configurations.

induce at the GGSN, and an evolved model. Furthermore, a differential analysis of our data compared to a newer dataset (potentially including LTE access) could really prove worthwhile.

In this paper we investigated trade-offs when virtualizing components of the mobile core network.

To this end, we first discussed a novel approach to mobile core network load modeling based on the control plane load at the GGSN. The $M/G/c_c$ loss model is based on the currently implemented state of the network architecture and can serve as a baseline reference to plan and dimension one's own mobile network accordingly, not just based on expected user traffic as traditionally. To improve scaling in the future, we proposed a completely new and virtualizable approach to a GGSN's makeup.

Then, we presented random variables to model load in a GGSN based on measurement data from the network of a nation-wide mobile service provider and made them available for reuse by other researchers.

Finally, we evaluate the model using a queing simulation. We have shown, that the system's blocking probability is roughly equal to the single-server model but in addition achieves large efficiency gains, even with very simplistic provisioning conditions and very long boot times. The model also has the ability to very easily scale out one's infrastructure by simply adding more small servers, reducing operational overhead. This might even lead to a new GGSN-as-a-Service business models, removing the need to provide and operate large amounts of infrastructure for rare cases of peak load. In the

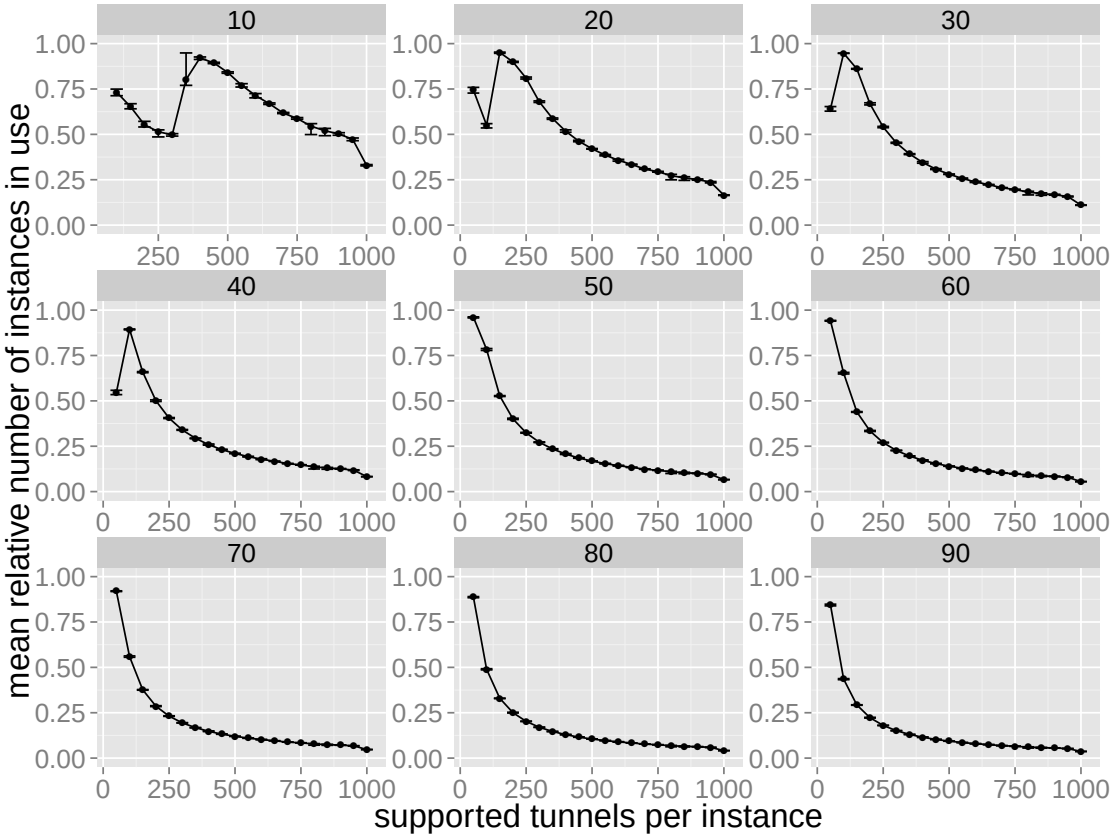


Figure 3.34: Mean instance usage of various server configurations.

future we would like to deepen our modeling efforts to provide more dimensioning options for a core network. Also, we want to further investigate the correlation of user traffic and signaling and take a look at the implications specific traffic types bring for the core network.

Table 3.3: Information Elements in a Create PDP Context Request (as peer TS 29.060, Section 7.3.1)

Information Element	**Presence Requirement**	**Typical Size**	**Information Element**	**Presence Requirement**	**Typical Size**
IMSI	Conditional	9	TFT	Conditional	min. 4
Routing Area Identity	Optional	7	Trigger Id	Optional	min. 4
Recovery	Optional	2	OMC Identity	Optional	min. 4
Selection mode	Conditional	2	Common Flags	Optional	4
Tunnel Endpoint Identifier Data I	Mandatory	5	APN Restriction	Optional	4
Tunnel Endpoint Identifier Control Plane	Conditional	5	RAT Type	Optional	4
Network Service Access Point Identifier (NSAPI)	Mandatory	2	User Location Information	Optional	11
Linked NSAPI	Conditional	2	MS Time Zone	Optional	5
Charging Characteristics	Conditional	3	IMEI(SV)	Conditional	11
Trace Reference	Optional	3	CAMEL Charging Information Container	Optional	min. 4
Trace Type	Optional	3	Additional Trace Info	Optional	12
End User Address	Conditional	9 (for IPv4)	Correlation-ID	Optional	4
Access Point Name	Conditional	3 + Name	Evolved Allocation Retention Priority I	Optional	4
Protocol Configuration Options	Optional	min. 3	Extended Common Flags	Optional	min. 4
SGSN Address for signaling	Mandatory	9 (for IPv4)	User CSG Information	Optional	11
SGSN Address for user traffic	Mandatory	9 (for IPv4)	APN-ABMR	Optional	11
MSISDN	Conditional	3 + MSISDN	Max MBR/APN-ABMR	Optional	min. 11
QoS Profile	Mandatory	min. 5	Private Extension	Optional	min 6
Signaling Priority Indication	Optional	min. 4			

Table 3.4: Relative TAC Statistics.

	Portion of devices with entry in TAC DB
# of Flows	99.72%
Ratio of Traffic	99.97%
# of Tunnels	87.57%
# of GTP Signaling Msgs	90.95%
# of Distinct MS-IDs	80.93%

Table 3.5: Parameters for the exponentially distributed inter-arrival times and corresponding Pearson correlation coefficients; also contains the inverse functions fitted to the empirical duration distribution and correlation coefficients of the fit.

Time of Day	λ	$R_{arrival}$	**Inverse Fitted Duration Function**	R_{dur}
oh-5h	10.67477	0.99538	$0.919208 - 60.6136y - 3498.78y^3 - \frac{110.707y + 2289.94y^3}{y - 1.00469}$	0.9999021
6h-11h	24.53298	0.99216	$1 + 117.484y - 368.643y^2 - \frac{1720.13y^4}{y - 1.0041}$	0.9998909
12h-17h	29.2504	0.99256	$0.952566 + 69.4907y + \frac{81146.1y^3 + 1.08572 \times 10^6 y^5}{805 - 802.01y}$	0.9999027
18h-23h	23.49983	0.98617	$0.911924 + 82.0562y - \frac{2936.93y^4}{1.94468y - 1.9532}$	0.9998071

Table 3.6: Typical abbreviation of processes in Kendall's notation.

Symbol	Description
M	Markovian, i.e. Poisson, arrival process or exponential service time distribution
D	Deterministic arrival process or service time distribution
G	General arrival process or service time distribution with no special assumptions
GI	General arrival process with independent arrivals; also called regenerative

Table 3.7: Manipulation check for the experimental factors based on one-way ANOVA.

	$F(2, 1275)$	η_p^2	p	Cohen's f^2	Cohen's $\hat{\omega}^2$
blocking probability					
maxTunnels	15601.534	0.993	< 0.001	26.739	0.964
maxInstances	10218.173	0.986	< 0.001	1.068	0.516
startstopDuration	0.868	0.003	0.482	0.000	0.000
mean number of tunnels					
maxTunnels	20448.347	0.994	< 0.001	27.712	0.965
maxInstances	13348.251	0.989	< 0.001	1.064	0.515
startstopDuration	2.872	0.009	0.022	0.000	0.000

4

STREAMING IN MOBILE NETWORKS

[TODO: Introduction, Background, and Related Work]

- LTE-EPC Network Simulator (LENA)[Bal11]
- Planetlab: an overlay testbed for broad-coverage services [Chu+03]
- SPDY: An experimental protocol for a faster web [BP11] and [BP12]
- An argument for increasing TCP's initial congestion window [Duk+10]
- Comparison of end-to-end and network-supported fast startup congestion control schemes [Sch11]
- Bufferbloat: Dark Buffers in the Internet [Get11]
- Detecting and quantifying bufferbloat in network paths [GK11]
- Congestion avoidance and control [Jac88]
- OsmocomBB project [WM11]
- Network neutrality, broadband discrimination [Wu03]
- Simulating LTE Cellular Systems: An Open-Source Framework [Pir+11]
- Controlling Queue Delay; Van Jacobson [NJ12]
- TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones. [Enc+10]
- An evaluation of dynamic adaptive streaming over HTTP in vehicular environments [MLT12]
- Firefox Patch: Sort Idle HTTP Connections by CWND [McM11]
- Characteristics of UDP packet loss: Effect of tcp traffic [Saw+]
- Improvements to congestion control for mobile streaming
- Alternative modes of transport (e.g. DCCP, ECN, ReECN, RED, Google's SPDY or new approaches)
- Influence of L1/2/3 layer protocols and mechanisms (IPv4/6, tunneling, ethernet vs RLC/RRC/NAS, ...)
- TCP modifications for short-lived connections (e.g. initially ignore congestion avoidance and push a lot of data at once (google.com approach, Increasing TCP's Initial Window))
- IW10: better response time, still fair http://code.google.com/speed/protocols/tcpm-IW10.html

4.1 PROPOSALS

4.2 UPCOMING PROTOCOLS AND STREAMING RELATIONSHIP

- SPDY / HTTP/2.0 (multiplexing -> segmented streaming)
- TCP changes (Fast Open, IW10, ...; any relationship to streaming? maybe faster)
- Alternate Transport Protocols (DCCP[KHF06a], LEDBAT[Sha+12b]/µtp[Nor], QUIC, SCTP, DTLS), HTTP/2.0 draft TODO
- Compensation mechanisms for reliable transport
- Model and quality estimations for improvements to adaptive streaming
- End-to-end encryption and Authentication mechanisms (e.g.IPSec, DNSSEC, CurveCP)
- Modifications to and issues with TCP
  - TCP buffer bloat
  - Initial window size (IW10, ...)
  - WebSockets as streaming transport [Hic13] [Weß11]
  - WebRTC
  - Relevance of multicasting or similar techniques for streaming transport (real-time live vs. stored)
  -

4.2.1 *Influence of Layers*

Wired Internet access has a very narrow choice of protocols on the ISO/OSI Layers 1 and 2. The typical use-case consists of a Local Area Network using Ethernet which is then tunneled through or translated into one of several access technologies, e.g., DSL, DOCSIS, or PON. Applications often make assumptions that rely on the presence of these protocols and their specific characteristics.

However, Internet access today is similarly often achieved using mobile cellular networks. The latest standardized iteration of these is LTE and the accompanying EPS core network infrastructure [Ols+09]. This is the first evolution of standards that completely removes the classical circuit switched domain making room for more radio frequency bandwidth to be used with the all-IP services achieving shared transmission capacities – comparable to today's 802.11n WiFi – albeit on much larger cell sizes of 1 to 30 kilometres. The EPS network (cf. Fig. fig:ltecore) acts as an intermediary between the radio access stations and the Internet enabling strong traffic control mechanisms as well as mobility anchored at the Serving Gateway (S-GW). Traffic is routed through the core by using tunneling over the S-GW and P-GW based on the traffic bearer concept defined either in the GTP or the PMIPv6 protocols. For every mobile device connected to the network there is one default and up to ten dedicated bearers carrying traffic filtered by pre-set QoS parameters. Control is enforced through a logically separate network control plane, that is also used to setup and tear down these bearers. Figure fig:ltestack

displays the disparity between the Internet's protocol stack and that of an LTE network encapsulating all user traffic in additional protocol layers by the tunneling process.

Research work is ongoing how to best work with this complex network setup. It is expected, that with the rise of mobile access the core network comes under heavy traffic pressure with negative affects on the QoS of best-effort traffic. Endeavors are required to study the loaded network's behavior. Also very little work has gone into exploring the control plane characteristics of these networks, including their performance. A novel approach could also be, to make mobile device applications, e.g. video streaming players, aware of the core networks capabilities and allow the request of tunnels tailored to their specific QoS requirements resulting in a possible increase of perceived quality.

The protocols used for the radio transmissions behave very differently when compared to Ethernet and assumptions made by higher layers may not hold any more. This can apply to, e.g., reliability, frame sizing and fragmenting, and latency amounting to undesired effects on higher-layer traffic. For example, loss in GSM and UMTS networks is often caught transparently on layer 2 and a retransmission is conducted. However, in the time the retransmission takes the transport layer may have already run into a timeout and re-requested the missing segment on its own, resulting in additional delay and a waste of bandwidth. This is especially detrimental for time-critical applications like video streaming, possibly resulting in buffer underruns and degraded quality. Transport and application layer mechanisms need to be able to understand this and cope with the effects. E.g., TCP retransmissions and congestion control could be adjusted in the course of understanding this.

Furthermore, traffic could be avoided during cell handover occurrences. This would require cross-layer cooperation and an awareness of the application when an handover is supposed to occur. The application then could schedule its traffic accordingly. Traffic falling into a handover is subject to especially high latency and loss because the mobile network acts as a mobility anchor which needs to internally reroute incoming traffic to the mobile device's new position. HTTP traffic is especially suited to this scheduling behavior because of its statelessness and consistence of small objects that can be requested and transferred independently.

4.2.2 *Cross-Layer Mobility Hinting*

i.e. Mobility without network support

Why? Handovers mean very long application interruption, and redirecting tcp flows to the current base station isn't cheap either (also, induces **very** long latency)

- Tell Transport/Application the expected time till handover / time of uninterrupted service
- Tell Transport/App expected connection parameters (latency, BW, ...)
- Application selects DASH stream appropriate to parameters
- Application reorders HTTP GETs so that large GETs are not interrupted
- Application stops transfers when handover is about to occur
- Layer 1/2 gives a list of possible handovers to Application

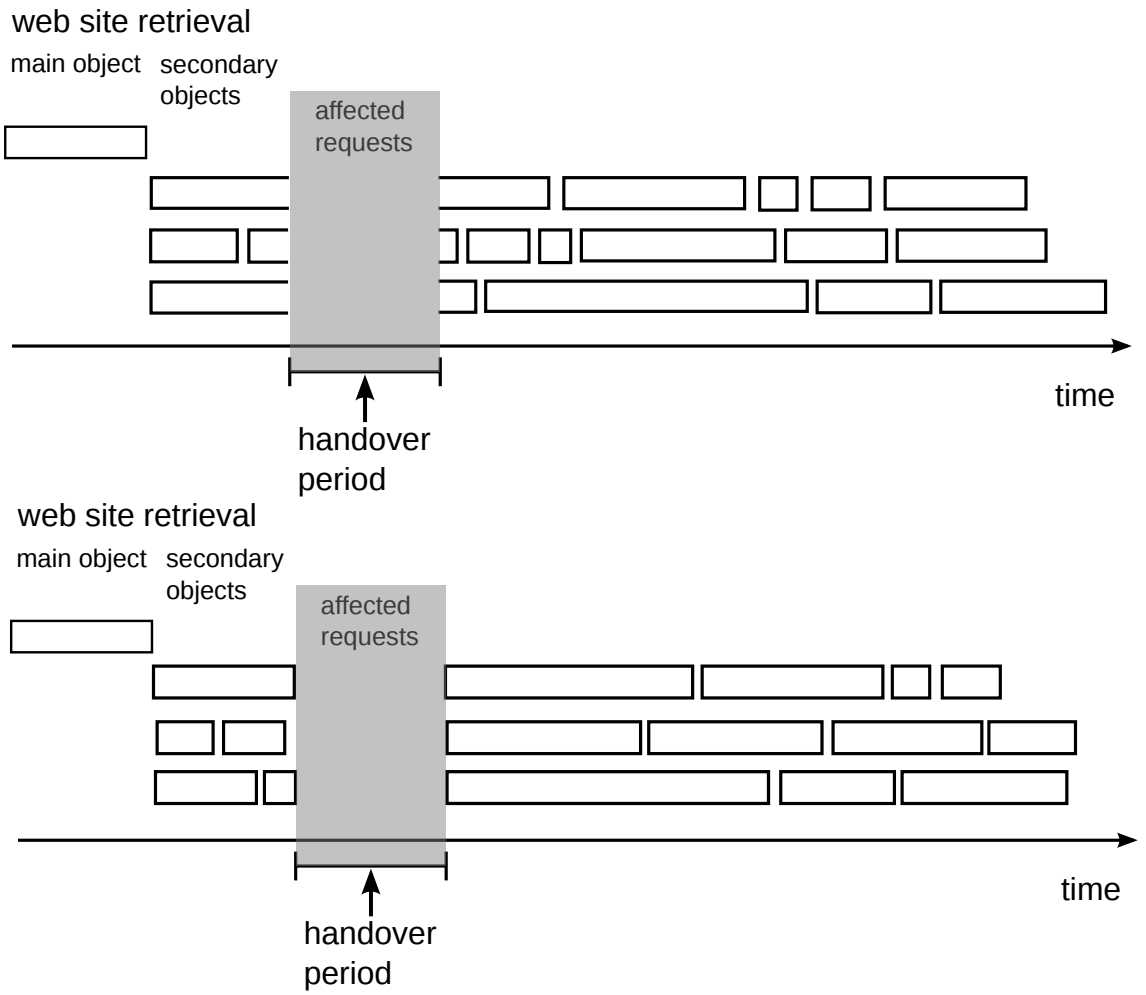


Figure 4.1: Mock-up of http reordering with handover awareness.

- Application selects (better: suggests) handover which fits best and reorders accordingly

Advantages/Disadvantages?
Violation of Layering/encapsulation? No, only APIs get extended
relevant work

- –> Cross-layer design optimizations in wireless protocol stacks [RI04a]
- ECLAIR: An efficient cross layer architecture for wireless protocol stacks [RI04b]
- Still no implementation of ECLAIR: "Cross-layer feedback architecture for mobile device protocol stacks" [RI06]
- 802.21
- A multi-layer mobility management architecture using cross-layer signalling interactions[WA03b]
- Dynamic Link Exchange Protocol (DLEP) [Rat+13]
- Seamless Mobility in Heterogeneous Wireless Networks [Zar+10]

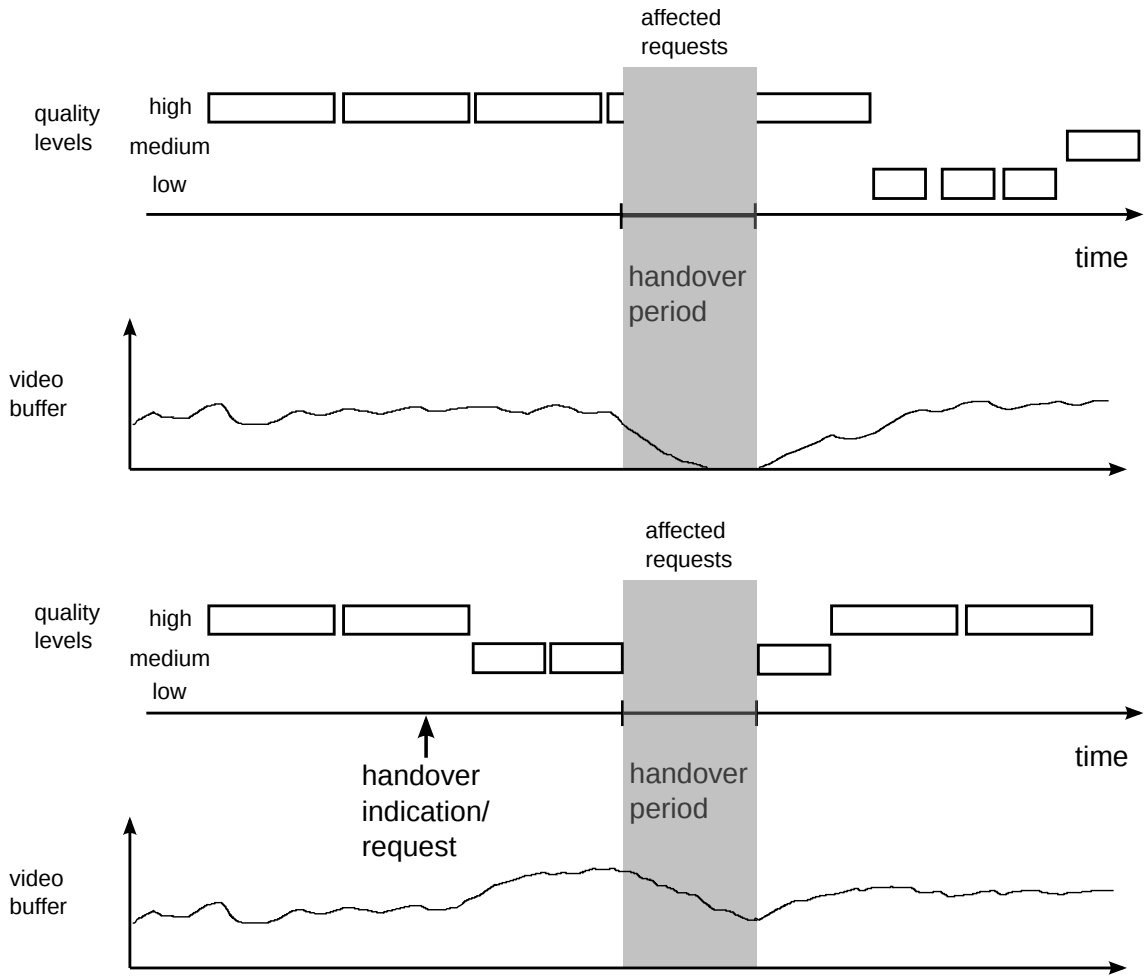


Figure 4.2: Mockup of handover prediction and hinting for adaptive streaming and thus avoiding playback stalls.

- Radio resource management in emerging heterogeneous wireless networks [Pia+11]
- Improved community network node design using a DLEP based radio-to-router interface [BR12]
- Cross-layer signalling for next-generation wireless systems [WA03a]
- Cross-layer design for wireless networks [SRK03]
- A cautionary perspective on cross-layer design [KK05]
- User-centric mobility management for multimedia content access [BRR11]
- Socketless TCP – an end to end handover solution [BV05]
- SATSIX cross-layer architecture [Mel+08]
- mostly routing protocol optimization oriented "A cross layer based QoS model for wireless and mobile ad hoc networks" [KI07]

SmoothIT mechanisms; lower layer elements provide information to higher layers, overlays [Oec+09]

Network Unassisted/Independent Mobility / Mobility Prediction API: When will the next handover occur, how much delay and bandwidth do i currently have? Can i complete oustanding transfers or should i reschedule them? Or, e.g., should i request a lower quality fragment? // Requirements: Stateless short to midtermed "Connections", high granularity of data "packets" –> "Mobility Awareness" [HHM10] PMLAR (Predictive mobility and location-aware routing protocol in mobile ad hoc networks)

4.3 INVESTIGATIONS

4.3.1 *Streaming Mobile Adaptation*

4.3.2 *Mobile Measurements with Additional Metadata*

SENSORIUM EXCURSION CONTEXT: (active) measurements in mobile networks should make use of the device's current state and environment. i.e., read its sensor data and put measurements in context with it. Alternatively, use anonymized sensor data for new kinds of evaluations (can this device watch video at all at the current location?) CONTEXT

Modern computing devices such as smartphones, laptops, and tablet computers are equipped with an increasing number of sensors: GPS, tilt, and acceleration meters quantify the physical position and orientation of the device; 3G, WiFi, and other interfaces gather data on the availability and signal quality of wireless networks; temperature and ambient light sensors deliver additional insight into a user's work and home environments.

There exist problems for the practical viability of collection data though: Different devices and platforms such as Android and iOS use very different interfaces into their sensors; privacy is another issue hardly tackled on any platform other than in a crude binary (allow/deny access) way. Therefore, in this demo we introduce Sensorium, a generic sensor reading framework that funnels data from actual sensor drivers, implements fine-grained privacy control for the user, and provides generic outbound interfaces such as XML-RPC. We also show an application using it, O3GM, which visualizes mobile coverage data coming from Sensorium.

Sensorium can access all the information a device provides and makes them available to other applications. Up until now, it has been a challenging task for software developers (especially scientists and experimenters) to implement specialized sensor applications. This task is simplified by providing a generic framework for interfacing sensors. In our current implementation, available for Android, most of the typical sensors are already implemented. Since giving access to sensor data also exposes the user's privacy, the user can disable or set privacy levels for each sensor individually, and all sensor readings that would be shared are displayed.

O3GM[1] showcases the sensors framework. It comprises a web service displaying cellular access technology data points at their GPS locations collected by devices running Sensorium (see Figure 4.3). This solves a real-world problem: Currently, this kind of data is only available to mobile operators, which however are hindered by commercial

1 http://homepage.univie.ac.at/albert.rafetseder/o3gm

interests to make them publicly available – at least in raw, unadorned form. Other projects such as OpenSignalMaps[2] and Sensorly[3], as well as corporations like Google and Apple collect these data, but are very restrictive regarding usage by other parties. This is not true for O3GM: We make the data points collected available as Open Data under an open content license.

Obviously, other applications are possible. Since code and data are open-sourced, everyone can implement their great ideas, port Sensorium to other platforms or reimplement our interface there, etc.

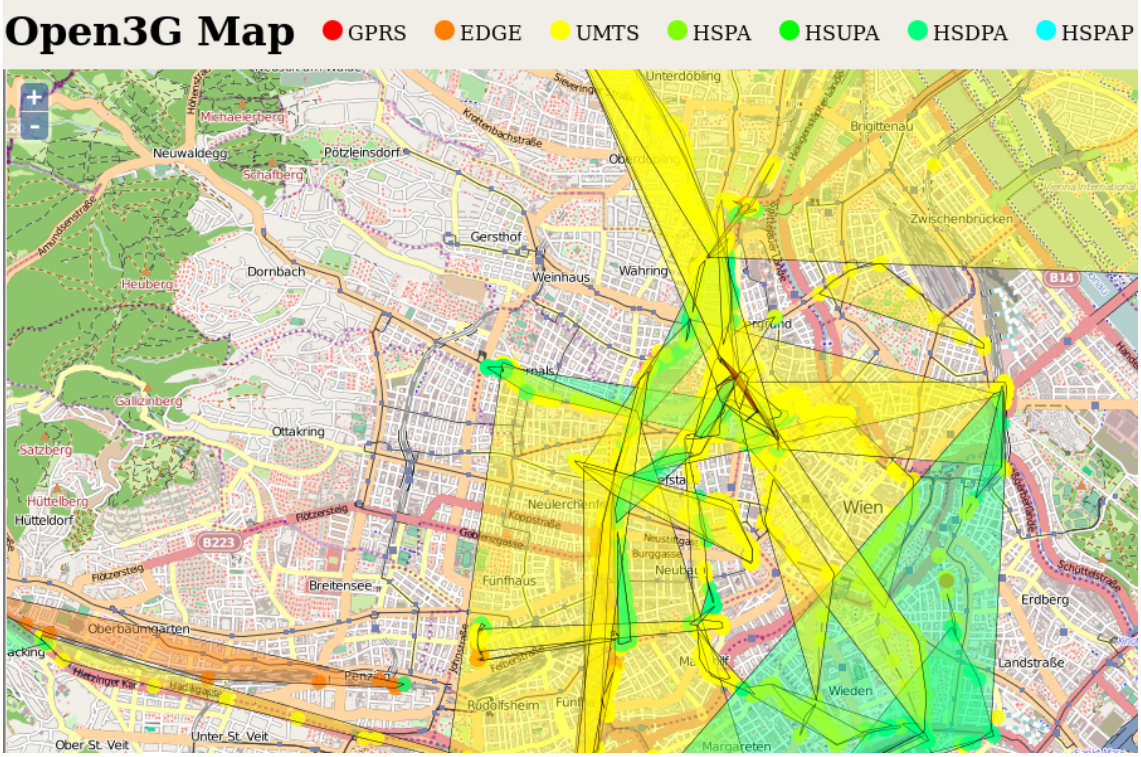


Figure 4.3: The O3GM web page, displaying 3G coverage measurements extracted from Sensorium on top of OpenStreetMap.

4.3.2.1 *Architecture*

Figure 4.4 overviews Sensorium's architecture components, and its interplay with the data collecting parts of O3GM. *Sensor drivers* are implemented on top of the operating system taking care of reading sensor values from platform-specific interfaces and pushing them upwards into the *registry*. Here, sensor data is timestamped and collected. On one side, data are prepared for local display, e.g. in a GUI or status widget. On the other side, a user-configurable *privacy layer* might allow for full sensor access from above or reduce the precision of values (e.g. round GPS coordinates); it could salt and hash sensor values for improved privacy, or completely deny access to individual (or all) sensors. Finally, other applications running on the same device are free to connect to Sensorium's *outbound interfaces* to register for sensor updates or poll data. Access from sources other than localhost is not allowed for obvious privacy reasons.

2 http://opensignal.com/
3 http://www.sensorly.com/

Due to the layered architecture, it is very simple for contributors to add their own implementations of layers or swap them out for their own altogether. Consider a scenario when a contributor wishes to include a sensor we do not yet provide a driver for. All that needs to be implemented is code interfacing the actual sensor, and the lightweight API into our sensor registry. Similarly, additional local display methods, privacy enhancements, and outbound interfaces might be implemented.

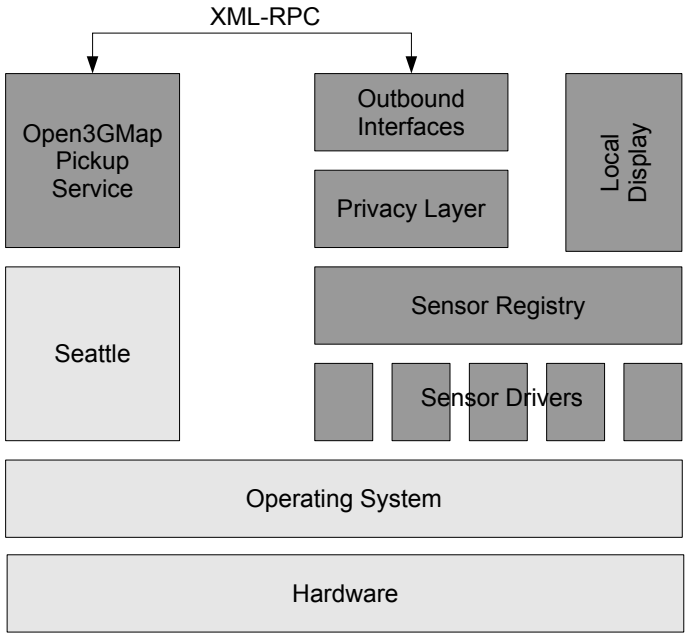


Figure 4.4: Sensorium architecture with O3GM *pickup* and *server*. Components described in this paper are shown dark gray.

4.3.2.2 *Implementation*

Our current implementation of Sensorium[4] runs on the Android platform. To provide a unified interface for accessing the sensor data we incorporated an XML-RPC library[5] that listens for connections on localhost, meaning that only applications running on the same device can access it. The pickup code[6] to collect sensor values runs on top of the renowned Seattle[7] platform, which is also available for Android, and allows us to remotely and securely access the collected data, and easily experiment with energy and cost efficient data upload/download strategies. The example application we implemented to make use of Sensorium, O3GM[8], is based on JavaScript and the OpenLayers[9] library. All of our code is dual-licensed under GPLv3 and the BSD license.

Sensorium consists of a base registry service with a common interface which each sensor implementation can easily be plugged into. All values are also displayed for the user as seen in Figure 4.5. The privacy layer automatically anonymizes all gathered

4 https://github.com/fmetzger/android-sensorium
5 https://code.google.com/p/android-xmlrpc/
6 https://homepage.univie.ac.at/albert.rafetseder/o3gm_pickup.repy
7 https://seattle.cs.washington.edu/
8 https://github.com/lukpueh/Open3GMap
9 http://openlayers.org/

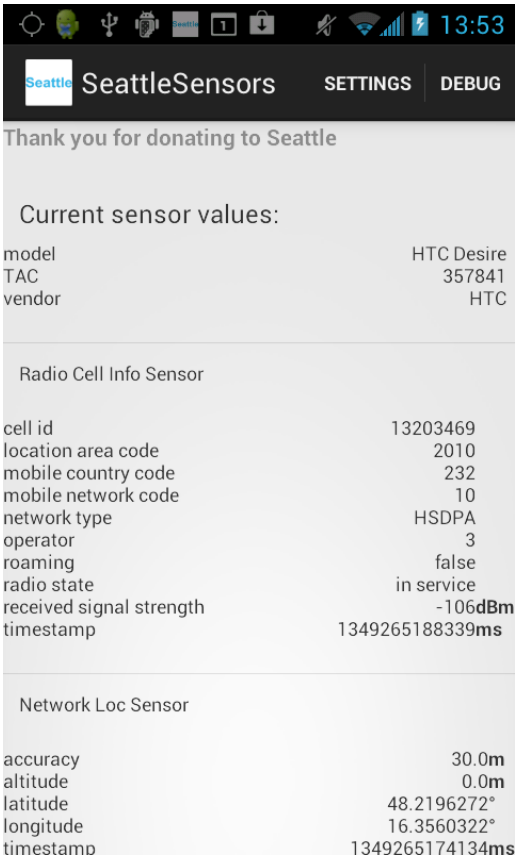


Figure 4.5: Sensorium screenshot.

values before making them available through XML-RPC in accordance with the user's current privacy settings.

We currently provide sensor implementations for generic device information, e.g. device name and battery status; mobile radio data such as current access technology and cell information; location data provided by the mobile network and GPS; and WiFi and Bluetooth information, including recent scan results.

Sensorium attempts to bring the sensing capabilities of current generation devices to a broader range of developers and experimenters. Directly using Sensorium, which is freely available from the Google Play store, or adopting the available source code gives everyone the chance to build projects like the mobile coverage Web service we presented.

4.3.3 *Measurement Approaches*

Different Approaches

4.3.3.1 *Simulating Streaming Traffic Characteristics in ns-3*

TODO: Implement a HTTP Streaming Traffic Generator ns-3 Application and a streaming receiver client application. TODO: Use LENA to transport streams TODO: Use Python TODO: Maybe base it on http://code.google.com/p/tmix-ns3/ ? TODO: Maybe use ON-OFF traffic generator with streaming specific patterns; cf. to http://www.nsnam.org/docs/rele

4.3.3.2 *Mobile Streaming Simulation/Emulation Hybrid Test Platform*

Use introduced streaming evaluation approaches in a mobile network environment

But do not want to use real network, as conditions are hard to manage and reproduce. Additionally, LTE still hard to come by.

Therefore, use an emulated network provided by the ns-3 network simulator. But transmit real traffic through it, i.e. use it as an emulator and bridges. 4.6

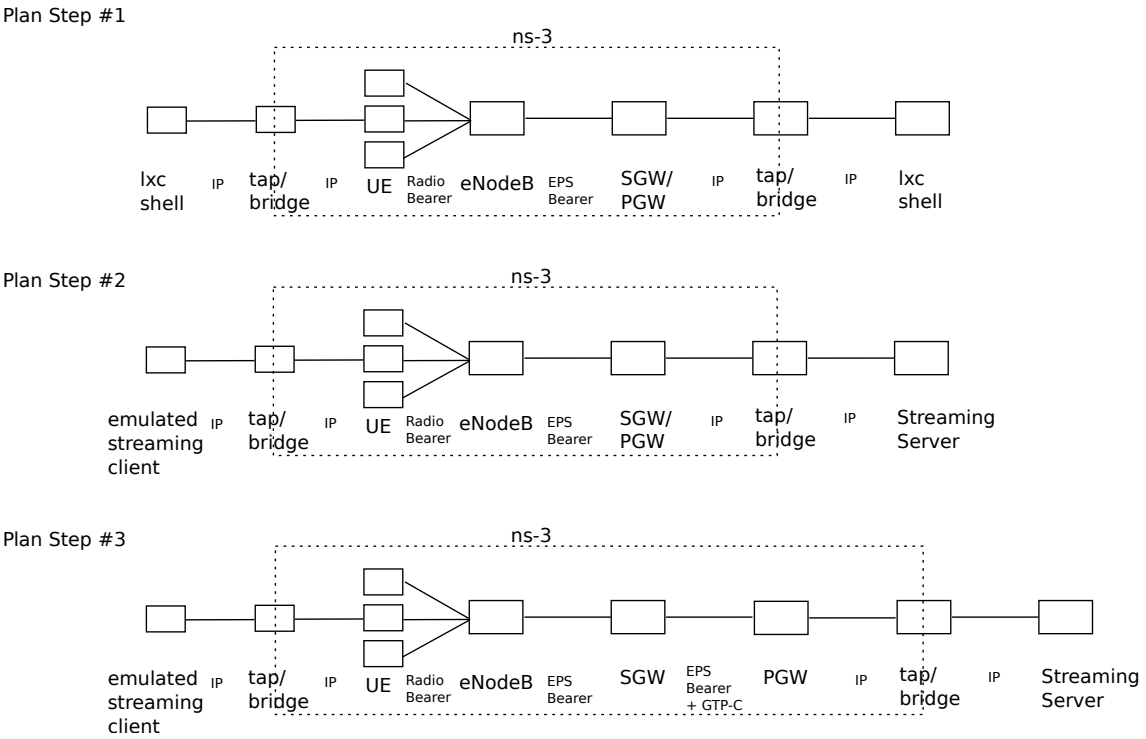


Figure 4.6: LTE Streaming Evaluation Setup and Action Plan

5

CONCLUSIONS

OWN PUBLICATIONS

[BMT12] A. Biernacki, **Metzger, F.**, and K. Tutschku. "On the Influence of Network Impairments on YouTube Video Streaming." In: *Journal of Telecommunications and Information Technology (JTIT)* 2012.03 (2012). URL: http://eprints.cs.univie.ac.at/3518/.

[Met+11] **Metzger, F.**, A. Rafetseder, D. Stezenbach, and K. Tutschku. "Analysis of web-based video delivery." In: *FITCE Congress (FITCE), 2011 50th*. 2011, pp. 1–6. DOI: 10.1109/FITCE.2011.6133425.

[Met+12] **Metzger, F.**, A. Rafetseder, P. Romirer, S. Gebert, K. Salzlechner, and K. Tutschku. "Research Report On Signaling Load and Tunnel Management in a 3G Core Network." In: (2012).

[MRR13] **Metzger, F.**, A. Rafetseder, and P. Romirer-Maierhofer. "Exploratory Data Analysis and Load Modeling of a 3G Core Network." In: *Journal of Computer Networks and Communications* (2013). Submitted.

[MRT12] **Metzger, F.**, A. Rafetseder, and K. Tutschku. "A performance evaluation framework for video streaming." In: *Packet Video Workshop (PV), 2012 19th International*. 2012, pp. 19–24. DOI: 10.1109/PV.2012.6229739.

[MSH14] **Metzger, F.**, C. Schwartz, and T. Hoßfeld. "A PDP Context Load Model and Virtualization Gain for a Mobile Network's GGSN." In: *17th International GI/ITG Conference on "Measurement, Modelling and Evaluation of Computing Systems" an"Dependability and Fault-Tolerance"', Bamberg 2014*. Submitted. 2014.

[Oec+09] S. Oechsner, F. Lehrieder, T. Hoßfeld, **Metzger, F.**, D. Staehle, and K. Pussep. "Pushing the performance of biased neighbor selection through biased unchoking." In: *Peer-to-Peer Computing, 2009. P2P'09. IEEE Ninth International Conference on*. IEEE. 2009, pp. 301–310.

[Raf+11] A. Rafetseder, **Metzger, F.**., D. Stezenbach, and K. Tutschku. "Exploring YouTube's Content Distribution Network Through Distributed Application-Layer Measurements: A First View." In: *Cnet 2011 : International Workshop on MODELING, ANALYSIS, AND CONTROL OF COMPLEX NETWORKS*. Sept. 2011.

[Raf+12] A. Rafetseder, **Metzger, F.**, D. Stezenbach, and K. Tutschku. "Network Federation as a Provider Concept: From Today's Measurement to Tomorrow's Architecture." In: *12th Würzburg Workshop on IP: ITG Workshop "Visions of Future Generation Networks" (EuroView2012)*. July 2012. URL: http://eprints.cs.univie.ac.at/3516/.

[RMP] A. Rafetseder, **Metzger, F.**, and L. Pühringer.
"Sensorium – A Generic Sensor Framework."
In: *PIK - Praxis der Informationsverarbeitung und Kommunikation* 36.1 (), p. 46.
DOI: doi:10.1515/pik-2012-0061.

BIBLIOGRAPHY

[13] *SimPy 3.0a1*. 2013. URL: https://simpy.readthedocs.org/.

[3GP07] 3GPP. *Customized Applications for Mobile network Enhanced Logic (CAMEL) Phase X; Stage 2*. TS 23.078.
3rd Generation Partnership Project (3GPP), Sept. 2007.
URL: http://www.3gpp.org/ftp/Specs/html-info/23078.htm.

[3GP08a] 3GPP. *Broadcast/Multicast Control (BMC)*. TS 25.324.
3rd Generation Partnership Project (3GPP), Jan. 2008.
URL: http://www.3gpp.org/ftp/Specs/html-info/25324.htm.

[3GP08b] 3GPP. *Evolved Universal Terrestrial Radio Access (E-UTRA) ; S1 Application Protocol (S1AP)*. TS 36.413.
3rd Generation Partnership Project (3GPP), Sept. 2008.
URL: http://www.3gpp.org/ftp/Specs/html-info/36413.htm.

[3GP08c] 3GPP. *IP Multimedia Subsystem (IMS); Stage 2*. TS 23.228.
3rd Generation Partnership Project (3GPP), Sept. 2008.
URL: http://www.3gpp.org/ftp/Specs/html-info/23228.htm.

[3GP08d] 3GPP. *Medium Access Control (MAC) protocol specification*. TS 25.321.
3rd Generation Partnership Project (3GPP), Sept. 2008.
URL: http://www.3gpp.org/ftp/Specs/html-info/25321.htm.

[3GP08e] 3GPP. *Multimedia Broadcast/Multicast Service (MBMS); Stage 1*. TS 22.146.
3rd Generation Partnership Project (3GPP), June 2008.
URL: http://www.3gpp.org/ftp/Specs/html-info/22146.htm.

[3GP08f] 3GPP. *Multimedia Broadcast/Multicast Service (MBMS) user services; Stage 1*. TS 22.246. 3rd Generation Partnership Project (3GPP), Mar. 2008.
URL: http://www.3gpp.org/ftp/Specs/html-info/22246.htm.

[3GP08g] 3GPP. *Radio Link Control (RLC) protocol specification*. TS 25.322.
3rd Generation Partnership Project (3GPP), Sept. 2008.
URL: http://www.3gpp.org/ftp/Specs/html-info/25322.htm.

[3GP08h] 3GPP. *UTRAN Iu interface Radio Access Network Application Part (RANAP) signalling*. TS 25.413.
3rd Generation Partnership Project (3GPP), Sept. 2008.
URL: http://www.3gpp.org/ftp/Specs/html-info/25413.htm.

[3GP11a] 3GPP. *Study on impacts on signalling between User Equipment (UE) and core network from energy saving*. TR 24.826.
3rd Generation Partnership Project (3GPP), June 2011.
URL: http://www.3gpp.org/ftp/Specs/html-info/24826.htm.

[3GP11b] 3GPP. *Study on non-MTC mobile data applications impacts*. TR 22.801.
3rd Generation Partnership Project (3GPP), Dec. 2011.
URL: http://www.3gpp.org/ftp/Specs/html-info/22801.htm.

[3GP12a] 3GPP. *General Packet Radio Service (GPRS); Service description; Stage 1*. TS 22.060. 3rd Generation Partnership Project (3GPP), Sept. 2012. URL: http://www.3gpp.org/ftp/Specs/html-info/22060.htm.

[3GP12b] 3GPP. *General UMTS Architecture*. TS 23.101. 3rd Generation Partnership Project (3GPP), Sept. 2012. URL: http://www.3gpp.org/ftp/Specs/html-info/23101.htm.

[3GP12c] 3GPP. *Packet Data Convergence Protocol (PDCP) specification*. TS 25.323. 3rd Generation Partnership Project (3GPP), Sept. 2012. URL: http://www.3gpp.org/ftp/Specs/html-info/25323.htm.

[3GP12d] 3GPP. *Physical layer - general description*. TS 25.201. 3rd Generation Partnership Project (3GPP), Sept. 2012. URL: http://www.3gpp.org/ftp/Specs/html-info/25201.htm.

[3GP12e] 3GPP. *Radio interface protocol architecture*. TS 25.301. 3rd Generation Partnership Project (3GPP), Sept. 2012. URL: http://www.3gpp.org/ftp/Specs/html-info/25301.htm.

[3GP12f] 3GPP. *Radio Resource Control (RRC); Protocol specification*. TS 25.331. 3rd Generation Partnership Project (3GPP), Sept. 2012. URL: http://www.3gpp.org/ftp/Specs/html-info/25331.htm.

[3GP12g] 3GPP. *UTRAN functions, examples on signalling procedures*. TR 25.931. 3rd Generation Partnership Project (3GPP), Sept. 2012. URL: http://www.3gpp.org/ftp/Specs/html-info/25931.htm.

[3GP13a] 3GPP. *3GPP Evolved Packet System (EPS); Evolved General Packet Radio Service (GPRS) Tunnelling Protocol for Control plane (GTPv2-C); Stage 3*. TS 29.274. 3rd Generation Partnership Project (3GPP), July 2013. URL: http://www.3gpp.org/ftp/Specs/html-info/29274.htm.

[3GP13b] 3GPP. *General Packet Radio Service (GPRS); GPRS Tunnelling Protocol (GTP) across the Gn and Gp interface*. TS 29.060. 3rd Generation Partnership Project (3GPP), Sept. 2013. URL: http://www.3gpp.org/ftp/Specs/html-info/29060.htm.

[3GP13c] 3GPP. *General Packet Radio Service (GPRS); Service description; Stage 2*. TS 23.060. 3rd Generation Partnership Project (3GPP), Sept. 2013. URL: http://www.3gpp.org/ftp/Specs/html-info/23060.htm.

[3GP13d] 3GPP. *GPRS Tunnelling Protocol for User Plane (GTPv1-U)*. TS 29.281. 3rd Generation Partnership Project (3GPP), Mar. 2013. URL: http://www.3gpp.org/ftp/Specs/html-info/29281.htm.

[3GP13e] 3GPP. *Mobile Application Part (MAP) specification*. TS 29.002. 3rd Generation Partnership Project (3GPP), Sept. 2013. URL: http://www.3gpp.org/ftp/Specs/html-info/29002.htm.

[3GP13f] 3GPP. *Mobile radio interface Layer 3 specification; Core network protocols; Stage 3*. TS 24.008. 3rd Generation Partnership Project (3GPP), Sept. 2013. URL: http://www.3gpp.org/ftp/Specs/html-info/24008.htm.

[3GP13g] 3GPP. *Network architecture*. TS 23.002. 3rd Generation Partnership Project (3GPP), June 2013. URL: http://www.3gpp.org/ftp/Specs/html-info/23002.htm.

[3GP13h] 3GPP. *Numbering, addressing and identification*. TS 23.003. 3rd Generation Partnership Project (3GPP), Sept. 2013. URL: http://www.3gpp.org/ftp/Specs/html-info/23003.htm.

[3GP13i] 3GPP. *Policy and charging control architecture*. TS 23.203. 3rd Generation Partnership Project (3GPP), Sept. 2013. URL: http://www.3gpp.org/ftp/Specs/html-info/23203.htm.

[3GP13j] 3GPP. *Study on Core Network (CN) overload solutions*. TR 23.843. 3rd Generation Partnership Project (3GPP), June 2013. URL: http://www.3gpp.org/ftp/Specs/html-info/23843.htm.

[3GP13k] 3GPP. *Study on GTP-C overload control mechanisms*. TR 29.807. 3rd Generation Partnership Project (3GPP), Sept. 2013. URL: http://www.3gpp.org/ftp/Specs/html-info/29807.htm.

[3GP13l] 3GPP. *Transparent end-to-end Packet-switched Streaming Service (PSS); Progressive Download and Dynamic Adaptive Streaming over HTTP (3GP-DASH)*. TS 26.247. 3rd Generation Partnership Project (3GPP), Sept. 2013. URL: http://www.3gpp.org/DynaReport/26247.htm.

[3GP13m] 3GPP. *Vocabulary for 3GPP Specifications*. TR 21.905. 3rd Generation Partnership Project (3GPP), June 2013. URL: http://www.3gpp.org/ftp/Specs/html-info/21905.htm.

[ABD11] S. Akhshabi, A. Begen, and C. Dovrolis. "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP." In: *Proceedings of the second annual ACM conference on Multimedia systems*. ACM. 2011, pp. 157–168.

[AJZ10] V. Adhikari, V. Jain, and V. Zhang. "YouTube traffic dynamics and its interplay with a tier-1 ISP: an ISP perspective." In: *Proceedings of the 10th annual conference on Internet measurement*. ACM. 2010, pp. 431–443.

[AN11] S. Alcock and R. Nelson. "Application flow control in YouTube video streams." In: *SIGCOMM Comput. Commun. Rev.* 41.2 (Apr. 2011), pp. 24–30. ISSN: 0146-4833. DOI: 10.1145/1971162.1971166. URL: http://doi.acm.org/10.1145/1971162.1971166.

[Aus10] Austinat, R. and Fechteler, P. and Gieselmann, H. "Über den Wolken: Wie Cloud Gaming den Spielemarkt revolutioniert." In: *c't* 21 (2010), pp. 76–83.

[BAB11a] A. Begen, T. Akgul, and M. Baugher. "Watching Video over the Web: Part 1: Streaming Protocols." In: *IEEE Internet Computing* 15.2 (Mar. 2011), pp. 54–63. ISSN: 1089-7801. DOI: 10.1109/MIC.2010.155. URL: http://dx.doi.org/10.1109/MIC.2010.155.

[BAB11b] A. Begen, T. Akgul, and M. Baugher. "Watching Video over the Web, Part II: Applications, Standardization and Open Issues."
In: *IEEE Internet Computing* 99.1 (2011). ISSN: 1089-7801.
DOI: 10.1109/MIC.2010.156.
URL: http://dx.doi.org/10.1109/MIC.2010.156.

[Bae+11] A. Baer, A. Barbuzzi, P. Michiardi, and F. Ricciato.
"Two parallel approaches to network data analysis."
In: *5th Workshop on Large Scale Distributed Systems and Middleware (LADIS)*. 2011.

[Bal11] N. e. Baldo. *LTE-EPC Network Simulator (LENA)*. 2011.
URL: http://iptechwiki.cttc.es/LTE-EPC_Network_Simulator_(LENA).

[Bar64] P. Baran. "On distributed communications networks."
In: *Communications Systems, IEEE Transactions on* 12.1 (1964), pp. 1–9.

[BC01] M. Blumenthal and D. Clark. "Rethinking the design of the Internet: the end-to-end arguments vs. the brave new world."
In: *ACM Transactions on Internet Technology (TOIT)* 1.1 (2001), pp. 70–109.

[BCZ97] S. Bhattacharjee, K. Calvert, and E. Zegura.
"Active networking and the end-to-end argument." In: *icnp*.
Published by the IEEE Computer Society. 1997, p. 220.

[Bel+13] M. Belshe, R. Peon, M. Thomson, and A. Melnikov.
Hypertext Transfer Protocol version 2.0. Internet-Draft.
Internet Engineering Task Force, Nov. 2013.
URL: http://tools.ietf.org/html/draft-ietf-httpbis-http2-08.

[Ber+13] R. Berjon et al., eds. *HTML5: A vocabulary and associated APIs for HTML and XHTML, section 4.8.10 Media Elements*. 2013.
URL: http://dev.w3.org/html5/spec/Overview.html#media-elements.

[BER11] S. Benno, J. O. Esteban, and I. Rimac.
"Adaptive streaming: The network HAS to help."
In: *Bell Labs Technical Journal* 16.2 (2011), pp. 101–114. ISSN: 1538-7305.
DOI: 10.1002/bltj.20505. URL: http://dx.doi.org/10.1002/bltj.20505.

[BMC04] J. Bannister, P. Mather, and S. Coope.
Convergence technologies for 3G networks: IP,UMTS, EGPRS and ATM.
John Wiley and Sons, May 2004.

[BP11] M. Belshe and R. Peon. *SPDY Protocol*. 2011.
URL: http://mbelshe.github.com/SPDY-Specification/.

[BP12] M. Belshe and R. Peon. *SPDY: An experimental protocol for a faster web*. 2012.
URL: http://www.chromium.org/spdy/spdy-whitepaper.

[BR12] C. Barz and H. Rogge. "Improved community network node design using a DLEP based radio-to-router interface."
In: *Wireless and Mobile Computing, Networking and Communications (WiMob), 2012 IEEE 8th International Conference on*. 2012, pp. 636–642.
DOI: 10.1109/WiMOB.2012.6379143.

[BRB08] A. Barbuzzi, F. Ricciato, and G. Boggia. "Discovering Parameter Setting in 3G Networks via Active Measurements."
In: *Communications Letters, IEEE* 12.10 (2008), pp. 730–732. ISSN: 1089-7798. DOI: 10.1109/LCOMM.2008.080913.

[BRR11] R. Bolla, R. Rapuzzi, and M. Repetto. "User-centric mobility management for multimedia content access." English.
In: *Multimedia Tools and Applications* (2011), pp. 1–29. ISSN: 1380-7501. DOI: 10.1007/s11042-011-0827-9.
URL: http://dx.doi.org/10.1007/s11042-011-0827-9.

[BT13] A. Biernacki and K. Tutschku.
"Comparative Performance Study of LTE Downlink Schedulers." English.
In: *Wireless Personal Communications* (2013), pp. 1–15. ISSN: 0929-6212. DOI: 10.1007/s11277-013-1308-4.
URL: http://dx.doi.org/10.1007/s11277-013-1308-4.

[Buj+09] A. Buja et al.
"Statistical inference for exploratory data analysis and model diagnostics."
In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 367.1906 (2009), pp. 4361–4383.
DOI: 10.1098/rsta.2009.0120. eprint: http://rsta.royalsocietypublishing.org/content/367/1906/4361.full.pdf+html.
URL: http://rsta.royalsocietypublishing.org/content/367/1906/4361.abstract.

[BV05] E. Beda and N. Ventura.
"Socketless TCP - an end to end handover solution."
In: *Networks, 2005. Jointly held with the 2005 IEEE 7th Malaysia International Conference on Communication., 2005 13th IEEE International Conference on*. Vol. 2. 2005, pages. DOI: 10.1109/ICON.2005.1635680.

[Chu+03] B. Chun et al. "Planetlab: an overlay testbed for broad-coverage services."
In: *ACM SIGCOMM Computer Communication Review* 33.3 (2003), pp. 3–12.

[Cis13] Cisco. *The Zettabyte Era—Trends and Analysis*. 2013.
URL: http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/VNI_Hyperconnectivity_WP.html.

[CLC10] Y. Chang, T. Lin, and P. Cosman.
"Network-based IP packet loss importance model for H.264 SD videos."
In: *Packet Video Workshop (PV), 2010 18th International*. 2010, pp. 178–185.
DOI: 10.1109/PV.2010.5706836.

[CM10] L. Cicco and S. Mascolo. "An Experimental Investigation of the Akamai Adaptive Video Streaming." In: *HCI in Work and Learning, Life and Leisure*. Ed. by G. Leitner, M. Hitz, and A. Holzinger. Vol. 6389.
Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, pp. 447–464. ISBN: 978-3-642-16606-8.
DOI: 10.1007/978-3-642-16607-5_31.
URL: http://dx.doi.org/10.1007/978-3-642-16607-5_31.

[Cor11] S. Corner. *Angry Birds + Android + ads = network overload*. 2011. URL: http://www.itwire.com/business-it-news/networking/47823-angry-birds-android-ads-network-overload.

[DC02] C. Demichelis and P. Chimento. *IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)*. RFC 3393 (Proposed Standard). Internet Engineering Task Force, Nov. 2002. URL: http://www.ietf.org/rfc/rfc3393.txt.

[DeG11] S. DeGroot, ed. *Sandvine's Spring 2011 Global Internet Phenomena Report Reveals New Internet Trends*. 2011. URL: http://www.sandvine.com/news/pr_detail.asp?ID=312.

[DMP11] L. De Cicco, S. Mascolo, and V. Palmisano. "Feedback control for adaptive live video streaming." In: *Proceedings of the second annual ACM conference on Multimedia systems*. MMSys '11. San Jose, CA, USA: ACM, 2011, pp. 145–156. ISBN: 978-1-4503-0518-1. DOI: 10.1145/1943552.1943573. URL: http://doi.acm.org/10.1145/1943552.1943573.

[Don02] S. Donnelly. "High precision timing in passive measurements of data networks." Dissertation. PhD thesis. New Zealand: Waikato University, June 2002.

[Don12] M. Donegan. *Android Signaling Storm Rises in Japan*. 2012. URL: http://www.lightreading.com/blog.asp?blog_sectionid=414&doc_id=216929&f_src=lrdailynewsletter.

[Duk+10] N. Dukkipati et al. "An argument for increasing TCP's initial congestion window." In: *ACM SIGCOMM Computer Communication Review* 40.3 (2010), pp. 26–33.

[Ell10] P. Ellis. *The essential guide to effect sizes: Statistical power, meta-analysis, and the interpretation of research results*. Cambridge University Press, 2010.

[Enc+10] W. Enck et al. "TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones." In: *OSDI*. Vol. 10. 2010, pp. 255–270.

[Erm+11] J. Erman, A. Gerber, K. K. Ramadrishnan, S. Sen, and O. Spatscheck. "Over the top video: the gorilla in cellular networks." In: *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*. IMC '11. Berlin, Germany: ACM, 2011, pp. 127–136. ISBN: 978-1-4503-1013-0. DOI: 10.1145/2068816.2068829. URL: http://doi.acm.org/10.1145/2068816.2068829.

[Faj+12] V. Fajardo, J. Arkko, J. Loughney, and G. Zorn. *Diameter Base Protocol*. RFC 6733 (Proposed Standard). Internet Engineering Task Force, Oct. 2012. URL: http://www.ietf.org/rfc/rfc6733.txt.

[Fie+99] R. Fielding et al. *Hypertext Transfer Protocol – HTTP/1.1*. RFC 2616 (Draft Standard). Updated by RFCs 2817, 5785, 6266, 6585. Internet Engineering Task Force, June 1999. URL: http://www.ietf.org/rfc/rfc2616.txt.

[FM11] I. Fette and A. Melnikov. *The WebSocket protocol*. 2011. URL: http://tools.ietf.org/html/draft-ietf-hybi-thewebsocketprotocol.

[Get11] J. Gettys. "Bufferbloat: Dark Buffers in the Internet." In: *Internet Computing, IEEE* 15.3 (2011), pp. 96–96.

[GHP08] J. Gustafsson, G. Heikkila, and M. Pettersson. "Measuring multimedia quality in mobile networks with an objective parametric model." In: *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*. Oct. 2008, pp. 405–408.

[Gil+07] P. Gill, M. Arlitt, Z. Li, and A. Mahanti. "Youtube traffic characterization: a view from the edge." In: *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*. ACM. 2007, pp. 15–28.

[GK11] D. Groenewegen and H. Kleppe. "Detecting and quantifying bufferbloat in network paths." In: (2011).

[Gro+96] A.-V. T. W. Group, H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. *RTP: A Transport Protocol for Real-Time Applications*. RFC 1889 (Proposed Standard). Obsoleted by RFC 3550. Internet Engineering Task Force, Jan. 1996. URL: http://www.ietf.org/rfc/rfc1889.txt.

[GSM12] GSM Association. *Fast Dormancy Best Practises*. 2012. URL: http://www.gsma.com/newsroom/1-0-network-efficiency-task-force-fast-dormancy-best-practices.

[HCH06] H. Holbrook, B. Cain, and B. Haberman. *Using Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Protocol Version 2 (MLDv2) for Source-Specific Multicast*. RFC 4604 (Proposed Standard). Internet Engineering Task Force, Aug. 2006. URL: http://www.ietf.org/rfc/rfc4604.txt.

[He+12] X. He, P. Lee, L. Pan, C. He, and J. Lui. "A Panoramic View of 3G Data/Control-Plane Traffic: Mobile Device Perspective." In: *Proceedings of IFIP/TC6 Networking 2012*. Proceedings of IFIP/TC6 Networking. Prague, Czech Republic, May 2012.

[Hem05] S. Hemminger. "Network emulation with NetEm." In: *Linux Conf Au*. Citeseer. 2005, pp. 18–23.

[HHM10] K. Hummel, A. Hess, and H. Meyer. "Mobilität im, Future Internet." In: *Informatik-Spektrum* 33.2 (2010), pp. 143–159.

[HHM11] H. Hisamatsu, G. Hasegawa, and M. Murata. "Non Bandwidth-intrusive Video Streaming over TCP." In: *Information Technology: New Generations (ITNG), 2011 Eighth International Conference on*. 2011, pp. 78–83. DOI: 10.1109/ITNG.2011.21.

[Hic13] I. Hickson, ed. *The WebSocket API, Editor's Draft*. 2013. URL: http://dev.w3.org/html5/websockets/.

[HJ98] M. Handley and V. Jacobson. *SDP: Session Description Protocol*. RFC 2327 (Proposed Standard). Obsoleted by RFC 4566, updated by RFC 3266. Internet Engineering Task Force, Apr. 1998. URL: http://www.ietf.org/rfc/rfc2327.txt.

[Hos+11] T. Hossfeld, M. Seufert, M. Hirth, T. Zinner, P. Tran-Gia, and R. Schatz. "Quantification of YouTube QoE via Crowdsourcing." In: *Multimedia (ISM), 2011 IEEE International Symposium on*. 2011, pp. 494–499. DOI: 10.1109/ISM.2011.87.

[Hos+12] T. Hossfeld, S. Egger, R. Schatz, M. Fiedler, K. Masuch, and C. Lorentzen. "Initial delay vs. interruptions: Between the devil and the deep blue sea." In: *Quality of Multimedia Experience (QoMEX), 2012 Fourth International Workshop on*. 2012, pp. 1–6. DOI: 10.1109/QoMEX.2012.6263849.

[HPW00] M. Handley, C. Perkins, and E. Whelan. *Session Announcement Protocol*. RFC 2974 (Experimental). Internet Engineering Task Force, Oct. 2000. URL: http://www.ietf.org/rfc/rfc2974.txt.

[Hua+12] T. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari. "Confused, timid, and unstable: picking a video streaming rate is hard." In: *Proceedings of the 2012 ACM conference on Internet measurement conference*. IMC '12. Boston, Massachusetts, USA: ACM, 2012, pp. 225–238. ISBN: 978-1-4503-1705-4. DOI: 10.1145/2398776.2398800. URL: http://doi.acm.org/10.1145/2398776.2398800.

[Ise97] D. Isenberg. "Rise of the stupid network." In: *Computer Telephony* 5.8 (1997), pp. 16–26.

[ITU04] ITU. *ITU-T Recommendation J.144: Objective perceptual video quality measurement techniques for digital cable television in the presence of a full reference*. Tech. rep. International Telecommunication Union, 2004. URL: http://www.itu.int/rec/T-REC-J.144/en.

[ITU08a] ITU. *ITU-T Recommendation G.1080: Quality of experience requirements for IPTV services*. Tech. rep. International Telecommunication Union, 2008. URL: http://www.itu.int/rec/T-REC-G.1080/en.

[ITU08b] ITU. *ITU-T Recommendation J.246: Perceptual visual quality measurement techniques for multimedia services over digital cable television networks in the presence of a reduced bandwidth reference*. Tech. rep. International Telecommunication Union, 2008. URL: http://www.itu.int/rec/T-REC-J.246/en.

[ITU08c] ITU. *ITU-T Recommendation J.247: Objective perceptual multimedia video quality measurement in the presence of a full reference*. Tech. rep. International Telecommunication Union, 2008. URL: http://www.itu.int/rec/T-REC-J.247/en.

[Jac88] V. Jacobson. "Congestion avoidance and control." In: *ACM SIGCOMM Computer Communication Review*. Vol. 18. 4. ACM. 1988, pp. 314–329.

[Jar+11] M. Jarschel, D. Schlosser, S. Scheuring, and T. Hoßfeld.
"An Evaluation of QoE in Cloud Gaming Based on Subjective Tests." In: *Workshop on Future Internet and Next Generation Networks (FINGNet-2011)*. Seoul, Korea, June 2011.

[Kaw+10] T. Kawano, K. Yamagishi, K. Watanabe, and J. Okamoto. "No reference video-quality-assessment model for video streaming services."
In: *Packet Video Workshop (PV), 2010 18th International*. 2010, pp. 158–164. DOI: 10.1109/PV.2010.5706833.

[Ken53] D. G. Kendall. "Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded Markov chain."
In: *The Annals of Mathematical Statistics* (1953), pp. 338–354.

[Ket+10] I. Ketykó et al. "QoE measurement of mobile YouTube video streaming."
In: *Proceedings of the 3rd workshop on Mobile video delivery*. MoViD '10. Firenze, Italy: ACM, 2010, pp. 27–32. ISBN: 978-1-4503-0165-7.
DOI: 10.1145/1878022.1878030.
URL: http://doi.acm.org/10.1145/1878022.1878030.

[KHF06a] E. Kohler, M. Handley, and S. Floyd.
Datagram Congestion Control Protocol (DCCP).
RFC 4340 (Proposed Standard). Updated by RFCs 5595, 5596, 6335, 6773. Internet Engineering Task Force, Mar. 2006.
URL: http://www.ietf.org/rfc/rfc4340.txt.

[KHF06b] E. Kohler, M. Handley, and S. Floyd.
"Designing DCCP: Congestion control without reliability." In: *ACM SIGCOMM Computer Communication Review* 36.4 (2006), pp. 27–38.

[KI07] P. Krishna and S. Iyengar.
"A cross layer based QoS model for wireless and mobile ad hoc networks."
In: *Mobile Communication* 1 (2007), pp. 114–120.

[KK05] V. Kawadia and P. Kumar.
"A cautionary perspective on cross-layer design."
In: *Wireless Communications, IEEE* 12.1 (2005), pp. 3–11. ISSN: 1536-1284.
DOI: 10.1109/MWC.2005.1404568.

[Kle75] L. Kleinrock. *Theory, Volume 1, Queueing Systems*. Wiley-Interscience, 1975. ISBN: 0471491101.

[KLL01] A. Klemm, C. Lindemann, and M. Lohmann.
"Traffic modeling and characterization for UMTS networks."
In: *Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE*. Vol. 3. 2001, 1741–1746 vol.3. DOI: 10.1109/GLOCOM.2001.965876.

[Knu97] D. E. Knuth.
The art of computer programming, volume 2 (3rd ed.): seminumerical algorithms. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1997. ISBN: 0-201-89684-2.

[Kol33] A. N. Kolmogorov.
"Sulla determinazione empirica di una legge di distribuzione."
In: *Giornale dell'Istituto Italiano degli Attuari* 4.1 (1933), pp. 83–91.

[KRA08] J. Kurose, K. Ross, and B. Anand. *Computer networking: a top-down approach*. Pearson/Addison Wesley, 2008.

[Lab+10] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, and F. Jahanian. "Internet inter-domain traffic."
In: *SIGCOMM Comput. Commun. Rev.* 41.4 (Aug. 2010), pages.
ISSN: 0146-4833.
URL: http://dl.acm.org/citation.cfm?id=2043164.1851194.

[Lan+12] M. Laner, P. Svoboda, P. Romirer-Maierhofer, N. Nikaein, F. Ricciato, and M. Rupp. "A Comparison Between One-way Delays in Operating HSPA and LTE Networks." In: *Proceedings of the 8th International Workshop on Wireless Network Measurements WinMee'12*.
Paderborn, Germany, May 2012.

[LBW07] P. Lee, T. Bu, and T. Woo.
"On the detection of signaling DoS attacks on 3G wireless networks."
In: *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*. IEEE. 2007, pp. 1289–1297.

[Lit61] J. D. Little. "A proof for the queuing formula: L= λW."
In: *Operations research* 9.3 (1961), pp. 383–387.

[LLoo] M. Lemley and L. Lessig. "End of End-to-End: Preserving the Architecture of the Internet in the Broadband Era, The."
In: *Ucla L. Rev.* 48 (2000), p. 925.

[Ma+11] K. Ma, R. Bartos, S. Bhatia, and R. Nair.
"Mobile video delivery with HTTP."
In: *Communications Magazine, IEEE* 49.4 (2011), pp. 166–175.
ISSN: 0163-6804. DOI: 10.1109/MCOM.2011.5741161.

[MCC11] R. Mok, E. Chan, and R. Chang.
"Measuring the quality of experience of HTTP video streaming."
In: *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*. 2011, pp. 485–492. DOI: 10.1109/INM.2011.5990550.

[McM11] P. McManus. "Firefox Patch: Sort Idle HTTP Connections by CWND." In: (2011). See also https://bugzilla.mozilla.org/show_bug.cgi?id=624739.
URL: http://bitsup.blogspot.com/2011/01/firefox-idle-connection-selection-via.html.

[Mee+03] J. van der Meer, D. Mackie, V. Swaminathan, D. Singer, and P. Gentric. *RTP Payload Format for Transport of MPEG-4 Elementary Streams*.
RFC 3640 (Proposed Standard). Updated by RFC 5691.
Internet Engineering Task Force, Nov. 2003.
URL: http://www.ietf.org/rfc/rfc3640.txt.

[Mel+08] I. Melhus et al. "SATSIX cross-layer architecture." In: *Satellite and Space Communications, 2008. IWSSC 2008. IEEE International Workshop on*. 2008, pp. 203–207. DOI: 10.1109/IWSSC.2008.4656786.

[MHM10] J. McAlarney, R. Haddad, and M. McGarry. "Modeling Network Protocol Overhead for Video." In: *Computer Modeling and Simulation (EMS), 2010 Fourth UKSim European Symposium on*. 2010, pp. 375–380. DOI: 10.1109/EMS.2010.68.

[MLT12] C. Müller, S. Lederer, and C. Timmerer. "An evaluation of dynamic adaptive streaming over HTTP in vehicular environments." In: *Proceedings of the 4th Workshop on Mobile Video*. MoVid '12. Chapel Hill, North Carolina: ACM, 2012, pp. 37–42. ISBN: 978-1-4503-1166-3. DOI: 10.1145/2151677.2151686. URL: http://doi.acm.org/10.1145/2151677.2151686.

[Mor+10] T. Mori, R. Kawahara, H. Hasegawa, and S. Shimogawa. "Characterizing traffic flows originating from large-scale video sharing services." In: *Traffic Monitoring and Analysis* (2010), pp. 17–31.

[MPE12] MPEG. *Dynamic adaptive streaming over HTTP (DASH) - Part 1: Media presentation description and segment formats*. International Standard 23009-1. ISO/IEC, 2012. URL: http://standards.iso.org/ittf/PubliclyAvailableStandards/c057623_ISO_IEC_23009-1_2012.zip.

[Nag84] J. Nagle. *Congestion Control in IP/TCP Internetworks*. RFC 896. Internet Engineering Task Force, Jan. 1984. URL: http://www.ietf.org/rfc/rfc896.txt.

[NFV13] NFV Industry Specification Group (ISG) in ETSI. *Network Functions Virtualisation – An Introduction, Benefits, Enablers, Challenges & Call for Action*. SDN & OpenFlow World Congress. Whitepaper. Oct. 2013.

[NJ12] K. Nichols and V. Jacobson. "Controlling queue delay." In: *Commun. ACM* 55.7 (July 2012), pp. 42–50. ISSN: 0001-0782. DOI: 10.1145/2209249.2209264. URL: http://doi.acm.org/10.1145/2209249.2209264.

[Nor] A. Norberg. *uTorrent transport protocol*. URL: http://bittorrent.org/beps/bep_0029.html.

[Ols+09] M. Olsson, S. Sultana, S. Rommer, L. Frid, and C. Mulligan. *SAE and the Evolved Packet Core: Driving the mobile broadband revolution*. Academic Pr, 2009.

[Pau+11] U. Paul, A. Subramanian, M. Buddhikot, and S. Das. "Understanding traffic dynamics in cellular data networks." In: *INFOCOM, 2011 Proceedings IEEE*. IEEE. 2011, pp. 882–890.

[PD07] L. Peterson and B. Davie. *Computer networks: a systems approach*. Morgan Kaufmann Pub, 2007.

[Pea00] K. Pearson. "X. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling." In: *Philosophical Magazine Series 5* 50.302 (1900), pp. 157–175. DOI: 10.1080/14786440009463897.

[Per+09] P. Perala, A. Barbuzzi, G. Boggia, and K. Pentikousis. "Theory and Practice of RRC State Transitions in UMTS Networks." In: *GLOBECOM Workshops, 2009 IEEE*. 2009, pp. 1–6. DOI: 10.1109/GLOCOMW.2009.5360763.

[Pia+11] K. Piamrat, A. Ksentini, J.-M. Bonnin, and C. Viho. "Radio resource management in emerging heterogeneous wireless networks." In: *Computer Communications* 34.9 (2011). <ce:title>Special Issue: Next Generation Networks Service Management</ce:title>, pp. 1066–1076. ISSN: 0140-3664. DOI: http://dx.doi.org/10.1016/j.comcom.2010.02.015. URL: http://www.sciencedirect.com/science/article/pii/S0140366410000903.

[Pir+11] G. Piro, L. Grieco, G. Boggia, F. Capozzi, and P. Camarda. "Simulating LTE Cellular Systems: An Open-Source Framework." In: *Vehicular Technology, IEEE Transactions on* 60.2 (Feb. 2011), pp. 498–513.

[PM13] R. Pantos and W. May. *HTTP Live Streaming*. Internet-Draft. 2013. URL: http://tools.ietf.org/html/draft-pantos-http-live-streaming-12.

[PP11] T. Porter and X.-H. Peng. "An Objective Approach to Measuring Video Playback Quality in Lossy Networks using TCP." In: *Communications Letters, IEEE* 15.1 (2011), pp. 76–78. ISSN: 1089-7798. DOI: 10.1109/LCOMM.2010.110310.101642.

[PT12] H. Parmar and M. Thornburgh. *Real-Time Messaging Protocol (RTMP) specification*. 2012. URL: https://www.adobe.com/devnet/rtmp.html.

[Qia+10] F. Qian, Z. Wang, A. Gerber, Z. Mao, S. Sen, and O. Spatscheck. "Characterizing radio resource allocation for 3G networks." In: *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. IMC '10. Melbourne, Australia: ACM, 2010, pp. 137–150. ISBN: 978-1-4503-0483-2. DOI: 10.1145/1879141.1879159. URL: http://doi.acm.org/10.1145/1879141.1879159.

[Qia+11] F. Qian, Z. Wang, A. Gerber, Z. Mao, S. Sen, and O. Spatscheck. "Profiling resource usage for mobile applications: a cross-layer approach." In: *Proceedings of the 9th international conference on Mobile systems, applications, and services*. ACM. 2011, pp. 321–334.

[Rao+11] A. Rao, A. Legout, Y. Lim, D. Towsley, C. Barakat, and W. Dabbous. "Network characteristics of video streaming traffic." In: *Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies*. CoNEXT '11. Tokyo, Japan: ACM, 2011, 25:1–25:12. ISBN: 978-1-4503-1041-3. DOI: 10.1145/2079296.2079321. URL: http://doi.acm.org/10.1145/2079296.2079321.

[Rat+13] S. Ratliff, B. Berry, G. Harrison, D. Satterwhite, and S. Jury. *Dynamic Link Exchange Protocol (DLEP)*. 2013. URL: http://tools.ietf.org/html/draft-ietf-manet-dlep-04.

[RCD10] F. Ricciato, A. Coluccia, and A. D'Alconzo. "A review of DoS attack models for 3G cellular networks from a system-design perspective." In: *Computer Communications* 33.5 (2010), pp. 551–558. ISSN: 0140-3664. DOI: 10.1016/j.comcom.2009.11.015. URL: http://www.sciencedirect.com/science/article/pii/S0140366409003168.

[RHR08] F. Ricciato, E. Hasenleithner, and P. Romirer-Maierhofer. "Traffic analysis at short time-scales: an empirical case study from a 3G cellular network." In: *Network and Service Management, IEEE Transactions on* 5.1 (2008), pp. 11–21. ISSN: 1932-4537. DOI: 10.1109/TNSM.2008.080102.

[RI04a] V. T. Raisinghani and S. Iyer. "Cross-layer design optimizations in wireless protocol stacks." In: *Computer Communications* 27.8 (2004). <ce:title>Advances in Future Mobile/Wireless Networks and Services</ce:title>, pp. 720–724. ISSN: 0140-3664. DOI: http://dx.doi.org/10.1016/j.comcom.2003.10.011. URL: http://www.sciencedirect.com/science/article/pii/S0140366403002913.

[RI04b] V. T. Raisinghani and S. Iyer. "ECLAIR: An efficient cross layer architecture for wireless protocol stacks." In: *WWC2004* (2004).

[RI06] V. Raisinghani and S. Iyer. "Cross-layer feedback architecture for mobile device protocol stacks." In: *Communications Magazine, IEEE* 44.1 (2006), pp. 85–92. ISSN: 0163-6804. DOI: 10.1109/MCOM.2006.1580937.

[Ric] F. Ricciato. "Introduction to the DARWIN project." In: (). Accessed: 06-Jul-2011. URL: http://userver.ftw.at/~ricciato/darwin/.

[Ric+06] F. Ricciato et al. "Traffic monitoring and analysis in 3G networks: lessons learned from the METAWIN project." In: *e & i Elektrotechnik und Informationstechnik* 123.7 (2006), pp. 288–296.

[Ros+02] J. Rosenberg et al. *SIP: Session Initiation Protocol*. RFC 3261 (Proposed Standard). Updated by RFCs 3265, 3853, 4320, 4916, 5393, 5621, 5626, 5630, 5922, 5954, 6026, 6141, 6665, 6878. Internet Engineering Task Force, June 2002. URL: http://www.ietf.org/rfc/rfc3261.txt.

[Ros+08] J. Rosenberg, R. Mahy, P. Matthews, and D. Wing. *Session Traversal Utilities for NAT (STUN)*. RFC 5389 (Proposed Standard). Internet Engineering Task Force, Oct. 2008. URL: http://www.ietf.org/rfc/rfc5389.txt.

[Ros09] P. Ross. "Cloud Computing's Killer App: Gaming." In: *Spectrum, IEEE* 46.3 (Mar. 2009), p. 14. ISSN: 0018-9235. DOI: 10.1109/MSPEC.2009.4795441.

[Ros10] J. Rosenberg. *Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols*. RFC 5245 (Proposed Standard). Updated by RFC 6336. Internet Engineering Task Force, Apr. 2010. URL: http://www.ietf.org/rfc/rfc5245.txt.

[Ros13] J. Ross, ed. *Sandvine Global Internet Phenomena Report 1H2013*. 2013. URL: http://www.sandvine.com/downloads/documents/Phenomena_1H_2013/Sandvine_Global_Internet_Phenomena_Report_1H_2013.pdf.

[RRCpt] P. Romirer-Maierhofer, F. Ricciato, and A. Coluccia. "Explorative analysis of one-way delays in a mobile 3G network." In: *Local and Metropolitan Area Networks, 2008. LANMAN 2008. 16th IEEE Workshop on*. Sept. Pp. 73–78. DOI: 10.1109/LANMAN.2008.4675847.

[Sai11a] P. Saint-Andre. *Extensible Messaging and Presence Protocol (XMPP): Core*. RFC 6120 (Proposed Standard). Internet Engineering Task Force, Mar. 2011. URL: http://www.ietf.org/rfc/rfc6120.txt.

[Sai11b] P. Saint-Andre. *Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence*. RFC 6121 (Proposed Standard). Internet Engineering Task Force, Mar. 2011. URL: http://www.ietf.org/rfc/rfc6121.txt.

[Saw+] H. Sawashima, Y. Hori, H. Sunahara, and Y. Oie. "Characteristics of UDP packet loss: Effect of tcp traffic." In: URL: https://www.isoc.org/inet97/proceedings/F3/F3_1.HTM.

[SBP13] M. Seyedebrahimi, C. Bailey, and X.-H. Peng. "Model and Performance of a No-Reference Quality Assessment Metric for Video Streaming." In: *CoRR* abs/1308.1839, abs/1308.1839 (2013).

[Sch+03] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. *RTP: A Transport Protocol for Real-Time Applications*. RFC 3550 (INTERNET STANDARD). Updated by RFCs 5506, 5761, 6051, 6222, 7022. Internet Engineering Task Force, July 2003. URL: http://www.ietf.org/rfc/rfc3550.txt.

[Sch+11] M. Schmidt, F. de Bont, S. Doehla, and J. Kim. *RTP Payload Format for MPEG-4 Audio/Visual Streams*. RFC 6416 (Proposed Standard). Internet Engineering Task Force, Oct. 2011. URL: http://www.ietf.org/rfc/rfc6416.txt.

[Sch+13] C. Schwartz, T. Hoßfeld, F. Lehrieder, and P. Tran-Gia. "Angry Apps: The Impact of Network Timer Selection on Power Consumption, Signalling Load, and Web QoE." In: *Journal of Computer Networks and Communications*, 176217 (2013). DOI: 10.1155/2013/176217.

[Sch11] M. Scharf. "Comparison of end-to-end and network-supported fast startup congestion control schemes." In: *Computer Networks* (2011).

[Sha+11] M. Shafiq, L. Ji, A. Liu, and J. Wang. "Characterizing and Modeling Internet Traffic Dynamics of Cellular Devices." In: *SIGMETRICS*. Vol. 11. 2011, pp. 305–316.

[Sha+12a] M. Z. Shafiq, L. Ji, A. X. Liu, J. Pang, and J. Wang.
"A first look at cellular machine-to-machine traffic: large scale measurement and characterization."
In: *Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE joint international conference on Measurement and Modeling of Computer Systems*. SIGMETRICS '12. London, England, UK: ACM, 2012, pp. 65–76.
ISBN: 978-1-4503-1097-0. DOI: 10.1145/2254756.2254767.
URL: http://doi.acm.org/10.1145/2254756.2254767.

[Sha+12b] S. Shalunov, G. Hazel, J. Iyengar, and M. Kuehlewind.
Low Extra Delay Background Transport (LEDBAT). RFC 6817 (Experimental).
Internet Engineering Task Force, Dec. 2012.
URL: http://www.ietf.org/rfc/rfc6817.txt.

[SHC12] R. Schatz, T. Hossfeld, and P. Casas.
"Passive YouTube QoE Monitoring for ISPs."
In: *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on*. 2012, pp. 358–364.
DOI: 10.1109/IMIS.2012.12.

[SHR12] K. Singh, Y. Hadjadj-Aoul, and G. Rubino. "Quality of experience estimation for adaptive HTTP/TCP video streaming using H.264/AVC."
In: *Consumer Communications and Networking Conference (CCNC), 2012 IEEE*. 2012, pp. 127–131. DOI: 10.1109/CCNC.2012.6181070.

[SL09] M. Schmidt and H. Lipson.
"Distilling Free-Form Natural Laws from Experimental Data."
In: *Science* 324.5923 (2009), pp. 81–85.

[SL13] M. Schmidt and H. Lipson. *Eureqa (Version 0.98 beta) [Software]*. 2013.
URL: http://www.eureqa.com/.

[SLT00] D. Staehle, K. Leibnitz, and P. Tran-Gia.
Source traffic modeling of wireless applications.
Inst. für Informatik, Universität Würzburg, 2000.

[Smi39] N. Smirnov. "On the estimation of the discrepancy between empirical curves of distribution for two independent samples."
In: *Bull. Math. Univ. Moscou* 2.2 (1939).

[SRC84] J. Saltzer, D. Reed, and D. Clark.
"End-to-end arguments in system design."
In: *ACM Transactions on Computer Systems (TOCS)* 2.4 (1984), pp. 277–288.

[SRK03] S. Shakkottai, T. Rappaport, and P. Karlsson.
"Cross-layer design for wireless networks."
In: *Communications Magazine, IEEE* 41.10 (2003), pp. 74–80. ISSN: 0163-6804.
DOI: 10.1109/MCOM.2003.1235598.

[SRL98] H. Schulzrinne, A. Rao, and R. Lanphier.
Real Time Streaming Protocol (RTSP). RFC 2326 (Proposed Standard).
Internet Engineering Task Force, Apr. 1998.
URL: http://www.ietf.org/rfc/rfc2326.txt.

[Sta+10] B. Staehle, M. Hirth, R. Pries, F. Wamser, and D. Staehle. "YoMo: A YouTube Application Comfort Monitoring Tool." In: *University of Wuerzburg, Tech. Rep* 467 (2010).

[Sta+11] B. Staehle, M. Hirth, R. Pries, F. Wamser, and D. Staehle. "Aquarema in action: Improving the YouTube QoE in wireless mesh networks." In: *Internet Communications (BCFIC Riga), 2011 Baltic Congress on Future*. 2011, pp. 33–40. DOI: 10.1109/BCFIC-RIGA.2011.5733220.

[Sto11] T. Stockhammer. "Dynamic adaptive streaming over HTTP –: standards and design principles." In: *Proceedings of the second annual ACM conference on Multimedia systems*. MMSys '11. San Jose, CA, USA: ACM, 2011, pp. 133–144. ISBN: 978-1-4503-0518-1. DOI: 10.1145/1943552.1943572. URL: http://doi.acm.org/10.1145/1943552.1943572.

[Svo+06] P. Svoboda, F. Ricciato, E. Hasenleithner, and R. Pilz. "Composition of GPRS, UMTS traffic: snapshots from a live network." In: *IPS MoMe 2006, Salzburg* 4 (2006), pp. 42–44.

[Tor+11] R. Torres, A. Finamore, J. R. Kim, M. Mellia, M. Munafo, and S. Rao. "Dissecting Video Server Selection Strategies in the YouTube CDN." In: *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*. 2011, pp. 248–257. DOI: 10.1109/ICDCS.2011.43.

[Tra96] P. Tran-Gia. *Analytische Leistungsbewertung verteilter Systeme - eine Einführung*. Springer, 1996, pp. I–XIII, 1–241. ISBN: 978-3-540-60666-6.

[Tut98] K. Tutschku. "Demand-based radio network planning of cellular mobile communication systems." In: *INFOCOM'98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*. Vol. 3. IEEE. 1998, pp. 1054–1061.

[WA03a] Q. Wang and M. Abu-Rgheff. "Cross-layer signalling for next-generation wireless systems." In: *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE*. Vol. 2. 2003, 1084–1089 vol.2. DOI: 10.1109/WCNC.2003.1200522.

[WA03b] Q. Wang and M. A. Abu-Rgheff. "A multi-layer mobility management architecture using cross-layer signalling interactions." In: (2003).

[Wan+03] B. Wang, J. Kurose, P. Shenoy, and D. Towsley. *A Model for TCP-based Video Streaming*. University of Massachusetts Technical Report. 2003. URL: http://lass.cs.umass.edu/papers/pdf/TR03-TCP.pdf.

[Wan+11a] Y.-K. Wang, R. Even, T. Kristensen, and R. Jesup. *RTP Payload Format for H.264 Video*. RFC 6184 (Proposed Standard). Internet Engineering Task Force, May 2011. URL: http://www.ietf.org/rfc/rfc6184.txt.

[Wan+11b] Z. Wang, Z. Qian, Q. Xu, Z. Mao, and M. Zhang.
"An untold story of middleboxes in cellular networks."
In: *Proceedings of the ACM SIGCOMM 2011 conference*. SIGCOMM '11.
Toronto, Ontario, Canada: ACM, 2011, pp. 374–385.
ISBN: 978-1-4503-0797-0.

[WC06] J. Welch and J. Clark. *A Proposed Media Delivery Index (MDI)*.
RFC 4445 (Informational). Internet Engineering Task Force, Apr. 2006.
URL: http://www.ietf.org/rfc/rfc4445.txt.

[WD09] S. Wang and S. Dey. "Modeling and characterizing user experience in a cloud server based mobile gaming approach."
In: *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*.
IEEE. 2009, pp. 1–7.

[Weß11] M. Weßendorf. "WebSocket: Annäherung an Echtzeit im Web." In: (2011).
URL: http://www.heise.de/developer/artikel/WebSocket-Annaeherung-an-Echtzeit-im-Web-1260189.html.

[WM11] H. Welte and S. Markgraf. *OsmocomBB project*. 2011.
URL: http://bb.osmocom.org/trac/.

[WSB03] Z. Wang, H. Sheikh, and A. Bovik. "Objective Video Quality Assessment."
In: *The Handbook of Video Databases: Design and Applications*.
Ed. by B. Furht and O. Marqure. CRC Press, 2003.

[Wu03] T. Wu. "Network neutrality, broadband discrimination."
In: *J. on Telecomm. & High Tech. L.* 2 (2003), p. 141.

[Xu+11] Q. Xu, J. Huang, Z. Wang, F. Qian, A. Gerber, and Z. Mao.
"Cellular data network infrastructure characterization and implication on mobile content placement."
In: *SIGMETRICS Perform. Eval. Rev.* 39.1 (June 2011), pp. 277–288.
ISSN: 0163-5999.

[ZÅ12] Y. Zhang and A. Årvidsson.
"Understanding the characteristics of cellular data traffic."
In: *SIGCOMM Comput. Commun. Rev.* 42.4 (Sept. 2012), pp. 461–466.
ISSN: 0146-4833. DOI: 10.1145/2377677.2377764.
URL: http://doi.acm.org/10.1145/2377677.2377764.

[Zar+10] F. Zarai, I. Smaoui, J.-M. BONNIN, L. Kamoun, et al.
"Seamless Mobility in Heterogeneous Wireless Networks."
In: *International Journal of Next-Generation Networks* 2.4 (2010).

[Zha+05] X. Zhang, J. Liu, B. Li, and T. Yum. "CoolStreaming/DONet: a data-driven overlay network for peer-to-peer live media streaming."
In: *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*. Vol. 3. 2005, 2102–2111 vol. 3.
DOI: 10.1109/INFCOM.2005.1498486.