universität wien

DISSERTATION

Titel der Dissertation

„Evaluating Reliable Streaming in Mobile 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:	Univ.-Prof. Dipl.-Ing. Dr. Helmut Hlavacs

© 2014 by Florian Metzger

This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.
https://creativecommons.org/licenses/by-sa/4.0/

Typeset in 11pt Libertine using LuaLaTeX.

CONTENTS

FIGURES

TABLES

ACRONYMS

μTP Micro Transport Protocol. 113
2G Second Generation. 23, 24
3G Third Generation. iv, vi, 11–13, 16, 19, 20, 23–26, 28–31, 33, 34, 36, 37, 39, 41–44, 48, 49, 71, 99, 111, 112, 121, 125, 133, 134, 149, 151
3GPP Third Generation Partnership Project. i, 11–13, 15, 25, 26, 29, 30, 45, 50, 84, 111, 134, 135, 146
ABI Application Binary Interface. 125
AMBR Aggregate Maximum Bit-Rate. 21
ANOVA Analysis of Variance. vii, 69, 73
API Application Programming Interface. 113, 115, 125, 130
APN Access Point Name. 19–21
AQM Active Queue Management. 112, 117
ARQ Automatic Repeat Request. 112
ATM Asynchronous Transfer Mode. 15
AU Access Unit. 79
AVC Advanced Video Coding. 102
BDP Bandwidth-Delay Product. 118, 136
BGP Border Gateway Protocol. 31
BMC Broadcast/Multicast Control. 15
BSS Base Station Subsystem. 13
CAMEL Customised Applications for Mobile Networks Enhanced Logic. 18, 21
CDF Cumulative Distribution Function. iv, v, x, 37–39, 45, 54–56
CDMA Code Division Multiple Access. 4
CDN Content Distribution Network. 83–85, 115
CELL_DCH Dedicated Channel. 24
CELL_FACH Forward Access Channel. 24
CELL_PCH Cell Paging Channel. 24
CGN Carrier-grade NAT. 27
CI Continuity Index. 88
CN Core Network. 11, 19, 24, 27, 28, 30, 31, 33, 41, 43, 47, 59
CS Circuit Switching. iv, 2, 4, 13, 14
CSG Closed Subscriber Group. 21
CSS Cascading Style Sheets. 1
DANE DNS-Based Authentication of Named Entities. 114
DASH Dynamic Adaptive Streaming over HTTP. 78, 83, 84, 95, 98, 150
DCCP Datagram Congestion Control Protocol. 78, 113, 114

DDoS Distributed DoS. 11, 28, 29
DES Discrete Event Simulation. 6, 58, 63, 107
DF Delay Factor. 88
DLEP Dynamic Link Exchange Protocol. 120
DNS Domain Name System. x, 83, 114
DNSSEC DNS Security Extensions. 114
DOCSIS Data Over Cable Service Interface Specification. 11, 111, 119
DoS Denial of Service. ix, 11
DPI Deep Packet Inspection. 16
DSL Digital Subscriber Line. 102, 111, 119
DTLS Datagram TLS. 114
E-UTRAN Evolved UTRAN. 13
ECDF Empirical CDF. iv, vii, 38, 39, 43–46, 50, 52–54, 56
ECN Explicit Congestion Notification. 119, 120
EDGE Enhanced Data Rates for GSM Evolution. x, 13
eNB Evolved Node B. vi, 134, 136, 139, 144, 145
EPC Evolved Packet Core. 15, 16, 19, 28, 30, 135, 151
EPS Evolved Packet System. 16
ES Elementary Stream. 79
ETSI European Telecommunications Standards Institute. 12
FEC Forward Error Correction. 114
FOSS Free and open-source software. 19, 134
FSM finite-state machine. 23
GERAN GSM/EDGE Radio Access Network. 13
GGSN Gateway GPRS Support Node. iv, v, 13, 16, 19, 20, 22, 23, 26, 27, 31–34, 42, 43, 47, 49, 51–53, 57, 59–66, 68, 70, 71, 73, 149–151, 157
GPL GNU General Public License. 157
GPRS General Packet Radio System. x, xiii, 4, 12, 13, 15, 16, 20, 22, 23, 26, 29, 37, 50–52, 111
GPS Global Positioning System. 34, 129
GSM Global System for Mobile Communications. iv, x, 4, 11–14, 23, 26, 47
GSMA GSM Association. i, 26, 29, 35
GTP GPRS Tunneling Protocol. i, ii, iv, x, xvii, xix, 16, 19, 20, 22–24, 28, 31–37, 41–43, 47, 49, 51–53, 59, 71, 111, 136, 146, 149, 151
GTP-C GTP Control. 20, 30
GTP-U GTP User. 19, 20, 23, 135
GUI Graphical User Interface. 131
HLR Home Location Register. 16, 32, 151
HLS HTTP Live Streaming. 83
HOL Head-of-line. 81
HSDPA High-Speed Downlink Packet Access. 13, 29
HSPA High Speed Packet Access. 4, 28, 50, 51

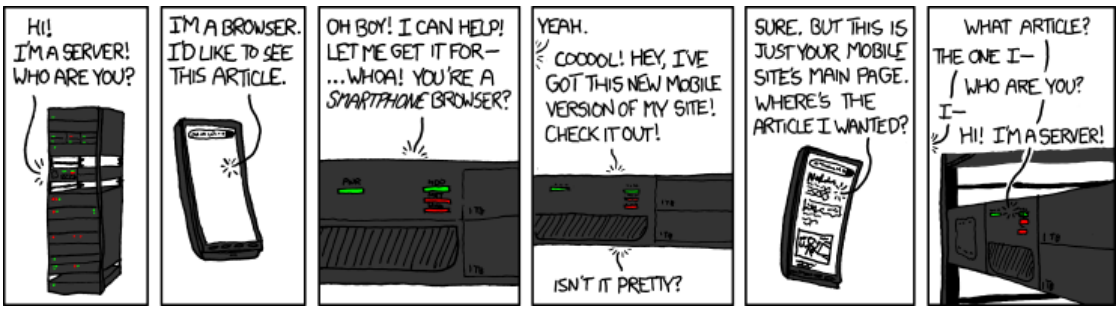
HSPA+ High Speed Packet Access Plus. 4, 13
HSS Home Subscriber Server. 16, 151
HSUPA High Speed Uplink Packet Access. 13
HTML HyperText Markup Language. 1, 94, 95, 116
HTTP HyperText Transfer Protocol. ix, xi, 27, 29, 34, 75, 78, 80–86, 93, 98, 100, 107, 110, 114–116, 118, 122, 124, 136, 150, 152
HTTPS HTTP Secure. 114, 115, 130
IAT Inter Arrival Time. iv, 49–55, 60, 63, 64, 88
ICE Interactive Connectivity Establishment. 80
IE Information Element. vii, 20–22, 33, 34, 47, 73, 151
IEC International Electrotechnical Commission. 83
IETF Internet Engineering Task Force. 7, 12, 25, 114, 116
IGMP Internet Group Management Protocol. 80, 84
IMAP Internet Message Access Protocol. xi
IMAPS IMAP Secure. 114
IMEI International Mobile Equipment Identity. 19, 21, 33, 35
IMS IP Multimedia Subsystem. 29, 84
IMSI International Mobile Subscriber Identity. 19–21, 34
IP Internet Protocol. iv, xi, xiii, 11–13, 15–17, 19, 29, 32, 42, 86, 109–111, 118, 119, 134, 135, 140
IPC Inter-Process Communication. 125
IPTV Internet Protocol television. 80, 88
IPv4 IP version 4. vii, 21
ISDN Integrated Services Digital Network. 4
ISO International Organization for Standardization. 7, 83, 119
ISP Internet Service Provider. 85, 86, 121
ITU International Telecommunication Union. 7, 87, 88
JSON JavaScript Object Notation. 130, 133
KISS “Keep it simple, stupid”. 151
LEDBAT Low Extra Delay Background Transport. 113
LGPLv3 GNU Lesser General Public License version 3. 157
LTE Long Term Evolution. iv, vi, xvii, 4, 11–15, 23, 28, 30, 50, 126, 127, 134–137, 139, 140, 144, 146, 147, 149–151
M2M Machine-to-Machine. 28, 36
MAC Media Access Control. 15, 120
MAP Mobile Application Part. 16
MBMS Multimedia Broadcast Multicast Services. 84
MDI Media Delivery Index. 88
METAWIN Measurement and Traffic Analysis in Wireless Networks. iv, 33–35
MIME Multipurpose Internet Mail Extensions. 34
MM Mobility Management. iv, 15, 23–25
MME Mobility Management Entity. 16, 134, 151

MOS Mean Opinion Score. vi, 86, 87, 89, 103, 105, 107, 142, 152
MS Mobile Station. 13, 21–24
MSE Mean Squared Error. 87
MSISDN Mobile Subscriber Integrated Services Digital Network-Number. 21
MTU Maximum Transmission Unit. xii, 111
NAT Network Address Translation. ix, xiii, 27, 80, 110
NFC Near Field Communication. 129
NFV Network Function Virtualization. v, 60, 61
NSAPI Network Service Access Point Identifier. 21
O3GM Open3G Map. vi, 133
OMC Operation and Maintenance Centre. 21
OS Operating System. iv, 1, 32, 36, 37, 41, 42, 44–46, 48, 64, 116, 118, 121, 152
OSI Open Systems Interconnection. 119
OSPF Open Shortest Path First. 31
P2P Peer-to-Peer. 84, 88
PCEF Policy and Charging Enforcement Function. 16
PCO Protocol Configuration Options. 21
PCRF Policy and Charging Rules Function. 16, 151
PDCP Packet Data Convergence Protocol. 15, 19, 111, 134
PDP Packet Data Protocol. i, iv, vii, 16, 18–23, 25, 26, 28–30, 42, 43, 146
PGW Packet Gateway. 16, 135, 136, 151
PI Pause Intensity. 88
PMTU Path MTU. 111
PON Passive Optical Network. 119
POTS Plain Old Telephone Service. 4
PPPoE Point-to-Point Protocol over Ethernet. 111
PRNG Pseudorandom Number Generator. 63
PS Packet Switching. iv, 13–15, 18, 22
PSNR Peak Signal-to-Noise Ratio. 87
QoE Quality of Experience. vi, 9, 27, 85–91, 103, 105, 124, 128, 139, 140, 142, 152
QoS Quality of Service. 6, 19, 21–23, 57, 75, 77, 82, 86, 88, 89, 92, 97, 99, 101, 107, 124, 128, 137, 139, 140, 147
QUIC Quick UDP Internet Connections. 114
RAB Radio Access Bearer. 16, 19, 23, 32, 111
RAI Routeing Area Identity. 21
RAN Radio Access Network. 19
RANAP Radio Access Network Application Part. 15, 16, 22
RAT Radio Access Technology. 21, 23, 26, 32–34, 47, 51, 133
RBI Reporting Body Identifier. 35
RFC Request for Comments. 25, 80
RIP Routing Information Protocol. 31

RLC Radio Link Control. 15, 111, 134
RNC Radio Network Controller. 15, 16, 23, 28, 29, 32, 33
RPC Remote Procedure Call. 130, 133
RRC Radio Resource Control. iv, 15, 22–25, 27–29, 44, 45, 47, 111, 134, 146
RTCP RTP Control Protocol. 78, 79
RTMP Real Time Messaging Protocol. 84
RTP Real-time Transport Protocol. xiii, 3, 78–81, 84, 114, 125, 149
RTSP Real Time Streaming Protocol. 78–80
RTT Round-Trip Time. 102, 111, 118
S1AP S1 Application Protocol. 16
SAP Session Announcement Protocol. 80
SAP Service Access Point. 119
SDP Session Description Protocol. 80
SDR Software Defined Radio. 12
SGSN Serving GPRS Support Node. iv, 13, 15, 16, 19–24, 27, 28, 32, 33, 42, 149, 151
SGW Serving Gateway. 16, 135, 136, 151
SIP Session Initiation Protocol. 29, 80
SMTP Simple Mail Transfer Protocol. xiii
SMTPS SMTP Secure. 114
SQL Structured Query Language. 35
STUN Session Traversal Utilities for NAT. 80
SV Software Version. 21
SVC Scalable Video Coding. 116, 152
TAC Type Allocation Code. ii, vii, 33–37, 43
TCP Transmission Control Protocol. vii, xvii, xix, 3, 8, 13, 15, 19, 28, 32, 75, 77–86, 88, 89, 99, 102, 104, 109–120, 124, 126, 134–138, 140, 149, 150, 152
TEID Tunnel Endpoint Identifier. 19–21
TFT Traffic Flow Template. 19, 21
TLS Transport Layer Security. x, 114–116
TS Technical Specification. 13, 18, 19, 24, 25, 30
TTI Transmission Time Interval. 50, 51
UDP User Datagram Protocol. xvii, xix, 20, 42, 78–81, 84, 110, 113, 114
UE User Equipment. 13, 16, 18–20, 34, 36, 44, 136, 139, 144
UMTS Universal Mobile Telecommunications System. iv, xiii, 4, 11–15, 18, 23, 29, 47, 50–52, 83, 111, 134, 149, 151
URA UTRAN Registration Area. xiii, 24
URA_PCH URA Paging Channel. 24
URL Uniform Resource Locator. 94, 95
UTRAN UMTS Terrestrial Radio Access Network. x, xiii, 13
VB Virtual Buffer. 88
VDSL Very-high-bit-rate Digital Subscriber Line. 11

VM Virtual Machine. 64, 127
VMM Virtual Machine Manager. 62
VoIP Voice over IP. 29, 124, 125
VQEG Video Quality Experts Group. 87
W3C World Wide Web Consortium. 95, 115
WebRTC Web Real-Time Communication. 115
XML Extensible Markup Language. 83, 130, 133
XMPP Extensible Messaging and Presence Protocol. 80

Server Attention Span



xkcd 869[1] by Randall Munroe, licensed under CC BY NC 2.5[2].

1 https://xkcd.com/869/
2 http://creativecommons.org/licenses/by-nc/2.5/

ABSTRACT

The usage of both video streaming as well as mobile networks has become prevalent today. Both make up a large portion of the Internet's current traffic mix. Additionally, video streaming has changed radically changed in recent years, shifting from traditional approaches running atop the unreliable UDP to using the reliable TCP as transport protocol. The behavior of these reliable streaming approaches can be much harder to predict because of the larger number of influence factors and cross-correlations between protocol layers. This is especially important in light of the complexity mobile networks are adding to the stack, even more so when the mobile control plane is factored in.

The purpose of this thesis is to evaluate this exact scenario: reliable streaming in mobile networks. The work takes a two-fold approach by first independently investigating both the mobile core network control plane as well as reliable streaming. For reliable streaming, a complete emulation-based measurement framework is introduced and measurement series that highlight the efficiency of certain playback strategies are conducted. Concerning mobile networks, a network trace from a live network is evaluated and explored for characteristics of the core network control plane. Through this, a network control plane load model based on GTP tunnel statistics is formulated and further evaluated through simulations.

Afterwards, the independent efforts are merged back together again with a study on the influences of network layers on streaming protocols. Many possible sources of influence on the quality of video streaming are discussed and a cross-layer information exchange framework, that can alleviate some of these negative factors, e.g., handover events in mobile networks, is presented.

Measuring anything related to this topic of reliable streaming in mobile networks is often a difficult task. But the efforts conducted here show that many of the factors related to it require a large amount of understanding and investigation. Therefore, the final part presents two further viable approaches to mobile streaming measurements: a mobile device active measurement environment that facilitates additional sensor and mobility data from the device, and lastly a reliable streaming simulation framework running atop a simulated LTE network to better evaluate mobility influences on streaming. The latter method is then ultimately used to show the effects of handover events on streaming.

All in all, this work gives insights into the relationships between reliable streaming and mobile networks. Measurement frameworks and models help to properly investigate them and deal with influencing factors surrounding the issue.

ZUSAMMENFASSUNG

Die Nutzung von Videostreaming als auch von Mobilfunknetzen ist in den letzten Jahren stark gestiegen und beide sind nun für einen Großteil des aktuellen Verkehrsmixes im Internet verantwortlich. Jedoch hat sich die Ausprägung des Videostreamings auch sehr stark im Vergleich zu früheren Formen geändert. Wo früher fast ausschließlich das unzuverlässige Transportprotokoll UDP zur Verwendung kam wird heute mehrheitlich auf das zuverlässige TCP gesetzt. Das Verhalten von zuverlässigen Streaming kann allerdings durch die größere Menge an Einflussfaktoren und Abhängigkeiten zwischen den Protokollschichten deutlich schwerer zu vorhersagen sein. Dieser Umstand fällt speziell bei Mobilfunknetzen und ihren Komplexen Protokoll- und Signalisierungsstrukturen ins Gewicht, insbesondere wenn zusätzlich auch die "Control Plane" der Mobilfunknetze betrachtet wird.

Genau dieses Szenario — Streaming mit zuverlässigen Transportprotokollen in Mobilfunknetzen — soll in dieser Dissertation vertieft untersucht werden. Dabei wird ein zweigleisiger Ansatz verfolgt: Zuerst werden Mobilfunknetze und zuverlässiges Streaming getrennt evaluiert. Für die Streaming-Untersuchung wird ein vollständiges emulationsbasiertes Messsystem eingeführt. Mit diesem werden Messreihen, die die Leistung von individuellen Abspielstrategien bewerten, durchgeführt. Mobilfunknetze werden insbesondere bezüglich ihrer Eigenschaften der Control Plane im Kernnetz untersucht. Dies erfolgt aufgrund eines Datensatzes eines Mitschnittes aus einem produktiven Mobilfunknetz und führt zu einer Definition von Last, die sich aus den Eigenschaften von GTP-Tunneln ableitet. Dieses Lastmodell wird dann durch statische Auswertungen und Simulationen weiter untersucht.

Anschließend verbinden sich diese beiden eigenständigen Forschungsansätze wieder in einer Studie zu den Einflussfaktoren von beispielsweise Netzwerkschichten auf Streaming-Protokolle. Hierbei werden viele Einflussquellen und auch ein möglicher Weg, diese durch einen Cross-Layer Informationsaustausch zu eliminieren oder zumindest zu reduzieren, diskutiert. Beispielsweise können durch diesen Ansatz die negativen Auswirkungen von Handover-Ereignissen auf Video-Streaming abgefangen werden.

Das Messen und Auswerten von zuverlässigen Mobil-Streaming ist in der Regel durch diverse Faktoren ein schwieriges Bestreben. Allerdings zeigen die hier vorgestellten Faktoren auch, dass eine große Menge an Untersuchungen nötig ist, um die Thematik hinreichend zu verstehen. Daher beleuchtet ein abschließender Abschnitt zwei weitere mögliche Messansätze: Einerseits eine aktive Messumgebung, die direkt auf einem mobilen Endgerät läuft und möglichst viele Sensorwerte und Protokollzustandsvariablen des Gerätes mit in eine Messung einbeziehen kann.Dies ist bei Mobilfunkmessungen, die auch häufig einer großen Mobilität und Einflussfaktoren durch den Benutzer unterliegen, ein essentieller Umstand. Zweitens wird eine zuverlässige Video-Streaming-Simulation auf einem vorhandenen Mobilfunksimulator implementiert und

dieser genutzt um exemplarisch den Einfluss von horizontalen Handovers auf eine laufende Streaming-Übertragung zu untersuchen.

Zusammenfassend gibt diese Arbeit Einblicke in die Beziehung zwischen zuverlässigen Streaming und Mobilfunknetzen. Die vorgestellten Mess- und Modellierungsmethodiken helfen bei der korrekten Analyse der vielen beobachtbaren Einflussfaktoren und führen letztlich zu einem besseren Verständnis der Thematik.

ACKNOWLEDGMENTS

I would like to thank my supervisor Professor Hlavacs for his advice and especially for taking over the supervision of my thesis at such a late stage on short notice. I would like to extend my thanks to my reviewers Professor Reichl as well as Professor Tran-Gia for accepting my review request and supporting me during the thesis process.

A big thanks also goes to my colleagues, former colleagues, as well as student helpers including Albert Rafetseder, David Stezenbach, Lukas Pühringer, Christoph Steindl, Katharina Salzlechner, Ákos Lukovics, Christian Schwartz, and Steffen Gebert for their general support, for their work on many big and small tasks, and last but not least for the educational and simultaneously entertaining discussions.

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 System (OS), especially including the release of UNIX in the early seventies.

From these early times countless other services and applications emerged. Today, the Internet is used by hundreds of millions of people and is involved in almost every aspect of daily life. The creation of the World Wide Web — initiated by Tim Berners-Lee in 1989 — 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, one of the first peer-to-peer file sharing services. However, without the large improvements in audio compression — the MP3 audio format in this case — a few years earlier, 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, video streaming and sharing sites like YouTube could not have existed without a wide spread availability of digital camcorders and other devices with recording capability, good video compression codes, and an even further increase in access bandwidths. For every advance in access bandwidth new services sprung to life fueling the users' demand for further capacity increases.

One of the driving factors in this rapid adoption of the Internet and the Web is its content agnosticism, i.e., that no assumptions are made on the transported data. The original idea is to treat every packet and participating node the same and not make any assumptions on specific applications. This provides a level playing field for every contender wanting to offer services. A further concept, called "End-to-End Principle", was originally stated as:

> "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. [...]" [SRC84]

This means that any functionality, for example services or applications, or traffic differentiation should only be implemented at the two endpoints of a transmission and not inside the network. Tuning the network to specific applications and implementing functions inside the network, will always only be valid for existing and narrow use cases and could hamper any future development.

During the past decades these principles have often been discussed [BCZ97; BC01; Ise97; LL00] but are mostly still being upheld in the current Internet topology. Interestingly, this is contrasted by the way Circuit Switching (CS) public telephone networks but also mobile networks operate. Here, much effort and intelligence is put into the network, which enables only a very specific set of services, determined solely by the operator, to work, albeit with a relatively high quality.

1.1 Video and the Internet

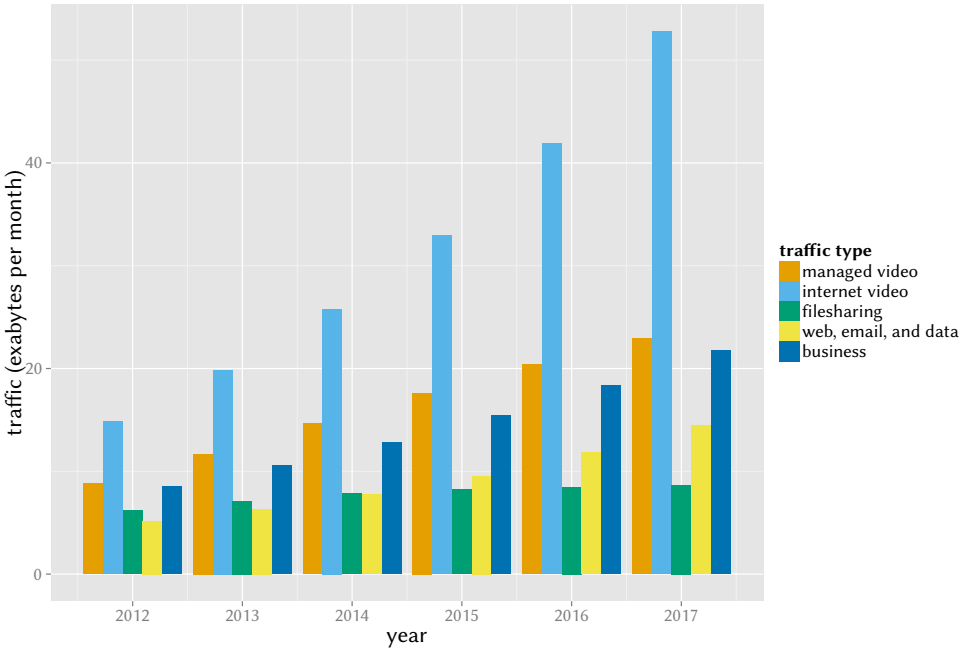


Figure 1.1: Cisco's global consumer Internet traffic prediction (data source: [Cis13]).

The total volume of Internet traffic has been rising rapidly for many years as most traffic studies suggest, for example in a Cisco study of "consumer" traffic in Figure 1.1. One of the largest contributing factors to today's traffic composition is arguably video traffic. It can be most dominantly observed in the mix of North America's fixed access depicted in Figure 1.2.

The two main sources for this video traffic are typically streaming services like YouTube[1] and Netflix[2], or live streaming and casting sites like Twitch[3].

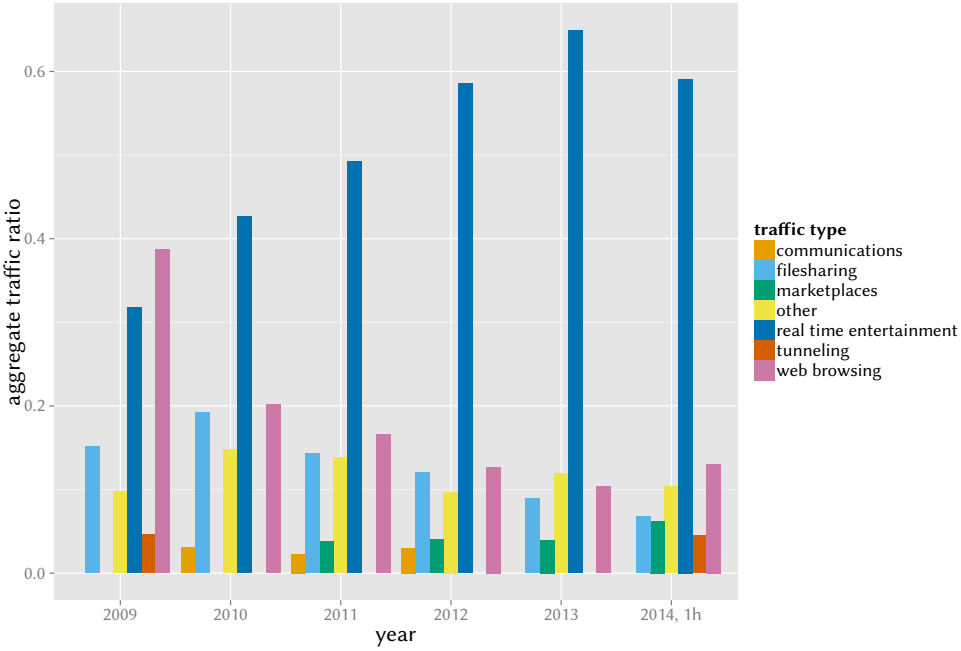


Figure 1.2: Traffic composition of North American peak fixed access aggregate traffic (data source: [San14]).

Studies predicting the development of the Internet's traffic also tell of the large influence of video traffic in the near future. As the video traffic volume has become so predominant, it is very important to understand its dynamics from a modeling and performance evaluation point of view, as specific behaviors of video traffic might significantly influence the underlying network and vice versa.

Protocols specifically created for the purpose of video streaming — which can be defined as "watching the video while later parts of it are still being transmitted" — have existed for a long time and their behavior is well researched. The most prominent is Real-time Transport Protocol (RTP), which has been specified as early as 1996 [RFC1889]. And yet they are not responsible for the gross of the Internet's video traffic seen today.

Instead, a new class of Transmission Control Protocol (TCP)-based and not formally standardized streaming approaches has arisen, which integrate themselves much better into the current Web ecosystem. And they are using completely different modes of transportation and control. Finding ways to investigate these new streaming protocols, including their categorization, measurement, and modeling, will be the first task of this thesis.

1 https://www.youtube.com
2 https://www.netflix.com
3 http://twitch.tv

Streaming, and media transport in general, is also not just relevant for the purpose of watching videos, there are many more fields of use with similar requirements. For example, so-called "cloud gaming" applications have become more popular in recent years. Here, video games are run on large virtualization clusters with the input and video output being streamed from and to the actual game client, putting a huge emphasis on achieving low latency. Refer to, e.g., [Ros09], [WD09], [Jar+11], or [Aus10] for further information on this topic.

1.2 The Mobile Network Below

Much of today's Web interactions are today carried out over mobile networks using smartphones. Despite their rapid evolution, mobile networks still carry much heritage from their circuit switching roots.[4] 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 Global System for Mobile Communications (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 the huge success of GSM, and its packet-switching extension General Packet Radio System (GPRS), its architecture was used as a blueprint for the following mobile network standard evolutions Universal Mobile Telecommunications System (UMTS) (including High Speed Packet Access (HSPA) and High Speed Packet Access Plus (HSPA+)) and Long Term Evolution (LTE).

Through this heritage, many mobile network elements and protocol have undergone only minor changes since their GSM versions. Due to the hereditary roots in CS networks, a large amount of state is kept inside the network and explicitly managed through a high volume of signaling interactions. This can create a wide range of problems, creating unwanted interactions of specific traffic patterns with the control plane and overwhelming the network's control plane structures.

These issues have only begun to show up in recent years due to the large influx of new users and usage scenarios. 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 faster access technologies many are now using their phones as the primary device for interacting with the Internet. Video, too, has grown to contribute large amounts of cellular traffic as Figure 1.3 depicts.

However, heavy traffic on a stateful network poses unique challenges to the architectural design of network protocols, and the performance and dimensioning of the network. This becomes even more pronounced 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

4 For an historical overview of mobile phone standards refer to https://en.wikipedia.org/wiki/History_of_mobile_phones.

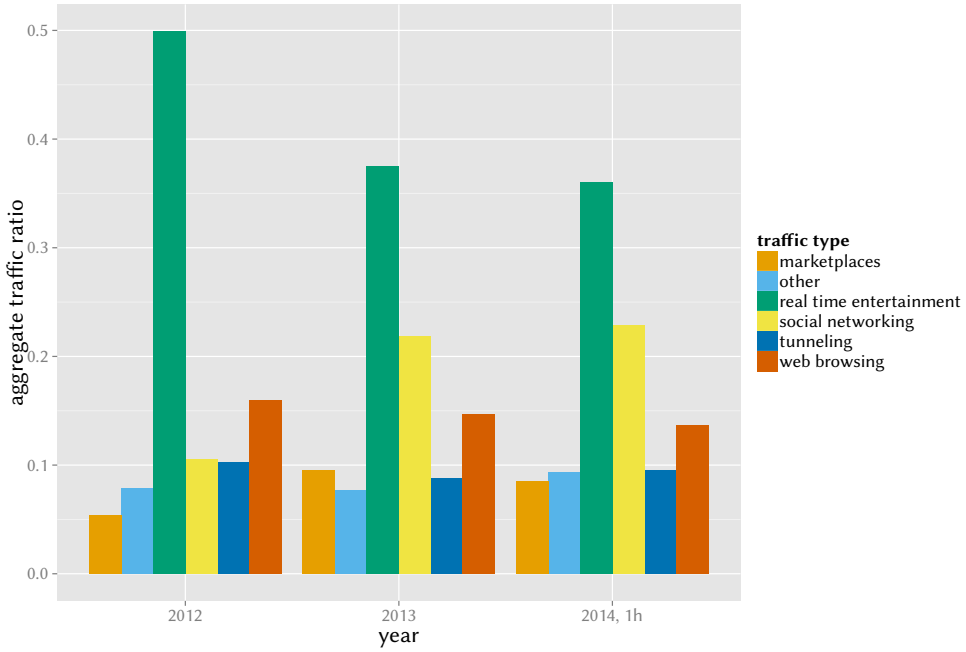


Figure 1.3: Traffic composition of North American peak mobile access aggregate traffic (data source: [San14]).

the exact make-up of these production networks making independent evaluation efforts rather difficult.

The investigation of the signaling structures in the core of today's mobile networks represents the second task of this thesis. It is also closely interlocked with the evaluation of video streaming traffic as they are nowadays often seen together.

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, tools are available to tackle most kinds of problems, but one still has to choose and properly define the most fitting ones for a given specific issue. Figure 1.4 categorizes 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.

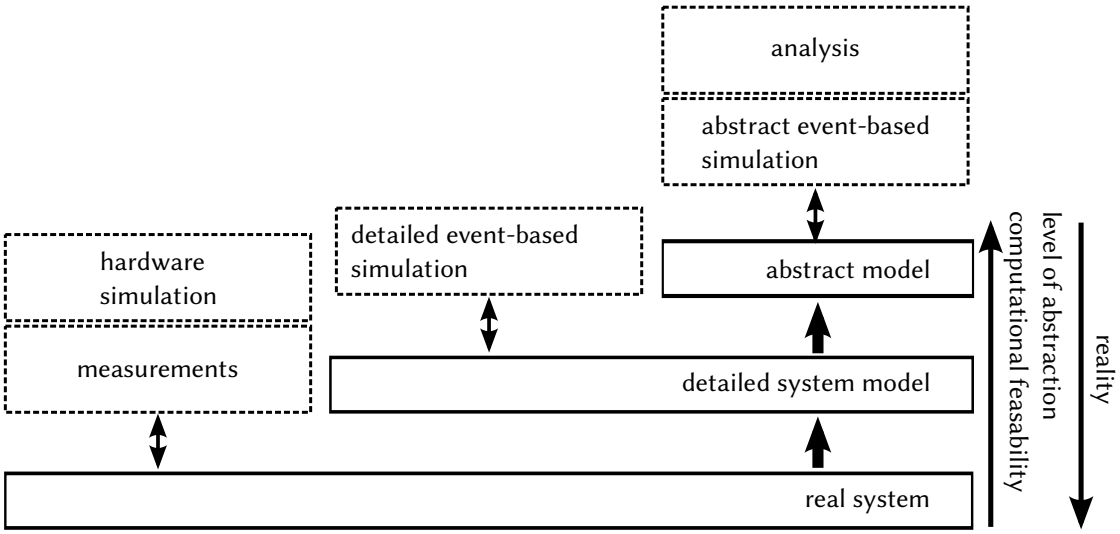


Figure 1.4: Methodical solution spaces and apparatus comparison (based on [Tra05, p. 2]).

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, most relevant to this type of research is the network emulator. Instead of having a real network, parts of it are substituted with emulating hardware or software that alters the Quality of Service (QoS) parameters of transmissions. 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 boundless, 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. A proper queuing theory model can already give great insights into the load dynamics of the modeled system and serve as a tool in future planning processes.

1.4 Research Contributions

With the increasing prevalence of video streaming in mobile networks, it is important to understand the relationship and interworking between these two. To achieve this with the

discussed tools at hand the topic will be separated into two lines of research discussing video streaming on the one hand and mobile networks on the other. After the individual issues have been thoroughly investigated on their own the efforts are merged again and actual video streaming in mobile networks is explored.

The first line of research will deal mostly with the control plane of mobile core networks. Using architectural knowledge, collected from the large stack of mobile network specifications, a measurement dataset from an actual core network is evaluated on the basis of its signaling load. Through further statistical analysis of specific signaling interactions a queuing theoretic load model of the core network well be created, abstractly describing the control plane.

This allows to draw interesting conclusions regarding the load state and its relationship to external influence factors, e.g., to user traffic, in these cellular nets. Building on top of this basis, an alternate model will be introduced that is better able to scale the load than the standard model. These efforts will ultimately help to improve the planning and dimensioning of networks, or even influence future specification iterations to reduce their adherence on signaling.

The second line will be an 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 quotation marks 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).

Also, their main distinction does not lie in the usage of the network and the video transport due to the reliance on typical Web protocols. Rather, much of the differences stems from the behavior of the actual application. Specifically, the way data is requested and retrieved and the video displayed, including its reaction to undesirable circumstances, e.g., having insufficient data in the playback buffer. Surprisingly, even just with these simple rules, applications can easily distinguish themselves from one another. Making the correct decisions to events can greatly influence the quality of the video playback.

During the investigation a model will be created that describes all the common elements in the streaming process. This model and additional knowledge of the structure and similarities of the applications enables further, more complex research efforts. For example, through a performance analysis streaming protocols can be compared to each other, or interrelations to the network's transport layer and other influences of the network on streaming could be uncovered.

This especially refers to influences of mobile networks, where many assumptions targeted at packet-switched wired networks do not hold anymore. One insightful example is the circumstance that the mobile network stack typically conceals any packet loss from its users. Loss will therefore only be experienced as intermittent phases of extremely high latency, which can go as high as several seconds, preventing transport protocols from correctly determining the link conditions and causing erroneous scaling. 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, this 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 third and final topic bridges both of the previous lines. The full video streaming mobile network stack is re-investigated on a quest for potential sources of influence that impact the quality of the resulting streaming endeavors. The search is made difficult by the circumstance that the stack is changing rapidly, with protocols being replaced or significantly altered. For example, the transmission behavior of the Linux implementation of TCP is being changed and improved upon constantly. Ultimately, an approach to reduce the stack's undesired influences will also be discussed in the form of cross-layer information flow.

Evaluating mobile networks and the complete streaming stack is a complex effort, with a vast number of variables to control and keep track of. To give an example, consider the mobility patterns of mobile devices, the resulting handover procedures between cells, and the effect on a video stream running during these events.

In this work, two methods are employed that deal with this situation. The first method enables one to conduct active measurements on mobile devices while combining them with the metadata from the device's stack and sensors. To achieve meaningful measurement results, the device's state needs to be recorded as precisely as possible. Today's mobile phones have a multitude of sensors and information sources available, which can be exploited for this.

The knowledge attained from these previous approaches can then finally be put into a mobile network streaming simulation, which can be more easily scaled up than an active measurement campaign. Through this method, new streaming strategies can be tried out and tested as well as tuned to the potential influences of the mobile network environment.

1.5 Structure

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

The facets of mobile core networks are put under scrutiny in both Chapters 2 and 3. As the amount of specifications as well as the details on the architecture, procedures, and protocols details is vast, the mobile core architectural background in Section 2.1 attempts to put all the basics together, with additional related work in Section 2.2.

With this foundation laid out, the actual goal of the investigations will now be defined in Section 2.3: a novel definition of load in a mobile network, based on the control plane and not just the user plane. To investigate this load a passive measurement data set is used. Section 2.4 describes the acquisition of the set and how it should be interpreted. The set is then evaluated for its signaling characteristics in Section 3.1 and the data is used to formulate two queuing theoretic load models in Section 3.2 which are then thoroughly tested using a queuing simulation

in Section 3.3. 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.

To introduce to the video streaming investigations, covered in Chapter 4, at first the protocols and techniques that have been used in the past will be described in Section 4.1. This then leads to a broad survey of current protocols. A subset of them will serve as the basis for the presented reliable streaming model.

Furthermore, to be able to derive Quality of Experience (QoE) information from the model, it is necessary to define an evaluation metric and its building blocks. In Section 4.3 these building blocks, that can be directly gathered from measurements, will be defined and an overview of existing metrics given. Now that both metric and model are ready, a measurement study using an emulation approach is conducted in Section 4.4, investigating the performance of an actual video streaming service with the help of the model under the influence of varying network conditions.

Chapter 5 includes the various efforts to screen and understand cross-layer influences with the discussion of current and upcoming protocols in Section 5.1 and a remedy in the form of a cross-layer hinting framework in Section 5.4.Finally, some thoughts and approaches to study video streaming traffic in mobile networks are collected in Chapter 6. Section 6.1 presents an active measurement approach that can collect and utilize additional metadata from the device. And Section 6.2 adapts the earlier described streaming measurement model to work with a mobile network simulation.

The thesis concludes with a summary, some remarks, and ideas for future work in Chapter 7.

2

MOBILE CORE NETWORK ARCHITECTURE

For the last few years, Internet access has been shifting towards mobile smartphones and mobile networks, sometimes even completely replacing fixed dial-up access with stationary mobile Third Generation (3G) and LTE modems. And all the "over-the-top" media streaming services are now more and more being used within mobile networks bringing a new set of traffic patterns as well as possible issues and influence factors along with them.

Today's cellular mobile networks are usually based on Third Generation Partnership Project (3GPP) specifications which have evolved from the circuit switched GSM network into the fully packet switched LTE, the latter still being in its unrolling phase. Yet being packet switched does not mean that mobile networks share a lot of similarity with a typical wireline Internet Protocol (IP) stack and network access 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 combined GSM and UMTS cellular network in use 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 implementing respective optional on-top services as is typically the case in the Internet. To achieve this, large amounts of state have to be kept at each network node and is explicitly communicated among the network nodes in its Core Network (CN). Also, the lines separating layers and functions are very blurred, making it very hard to grasp parts of the architecture without understanding the whole. Many specialized protocols are involved to communicate intents and states in the network. This causes processing overhead and signaling traffic on the network paths on top of the actually useful user traffic.

It is a known fact that radio spectral resources are a scarce resource that need to be well managed. But it might not seem immediately obvious to think the same about control plane resources in the infrastructure of a mobile network. Yet, there have been accounts[1][2][3] [Yan11] of situations where the core network has been flooded with signaling, despite only a negligible amount of actual user traffic being transported. This resulted in an event dubbed "signaling storm", causing an unintentional Distributed DoS (DDoS) attack, disrupting user-plane connectivity on its way. Similar patterns can also occur with some segmented transmission strategies in streaming.

The state of research on the architecture and especially the control plane behavior of 3G mobile cellular networks is lean in comparison to the huge volume of research conducted other wired Internet structures. Most of the available research focuses on user-oriented metrics

1 "Docomo Counts Cost of Signaling Storm",http://www.lightreading.com/d/d-id/693779
2 "Android Signaling Storm Rises in Japan", http://www.lightreading.com/a/d-id/693138
3 "Angry Birds + Android + ads = network overload", http://www.itwire.com/business-it-news/47823

such as traffic statistics and mobility patterns, or only investigates properties of the radio interface. Little has been published about activity within the core network, and yet less about core signaling.

Conducting IP traffic research at the network's edge, independent of the access technology, is usually relatively easy. Writing simple tests and measurement scripts, that record the user's traffic, is all that is needed. But mobile phones do not allow one to peek inside its layer 1 and 2 implementation and interaction, where the control plane resides. Any information on this black box can only be indirectly inferred from layers above — by forcing behavior known from specifications — or below — through spectrum analysis using Software Defined Radio (SDR) approaches.

However, to directly investigate the core as well as the control plane, some kind of a measurement infrastructure inside the network is needed. With this, researchers could not just look into user traffic flowing through the network but also investigate the signaling heavy mobile network control plane. But to gain this kind of access to the network would require the cooperation of a mobile network operator, which they grant very reluctantly due to privacy and competitive concerns.

Another important aspect related to signaling is network dimensioning. Operators mostly dimension their networks in accordance to the expected user traffic. But in such a signaling-dependent architecture this might not be the most fitting approach, as the mentioned signaling storms seem to demonstrate.

Both this and the next chapter attempt to remedy parts of the problem. The current chapter contains material required for a basic understanding of mobile networks and the evaluation methodology. Section 2.1 discusses the required details of the mobile network architecture under investigation followed by a survey of existing literature in Section 2.2. Next, an attempt on defining control plane load is made in Section 2.3. This is used to find viable targets for a load evaluation. The evaluation methodology is then given in Section 2.4.

This leaves Chapter 3 free to exclusively discuss the actual control plane modeling and evaluations conducted in an existing mobile network. Both chapters also use material previously published in [F M+12], [F M+14], and [FSH14] and extend on it.

2.1 Core Architecture Overview

Today's dominating commercial mobile network system, which combines GSM, UMTS, and now often also 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), *Release 6* (2004), or *Release 7* (2007) as they introduced High-Speed Downlink Packet Access (HSDPA), High Speed Uplink Packet Access (HSUPA), and HSPA+ respectively. LTE first found its way into the specifications in 2008 with *Release 8*. *Release 12* is scheduled to be published in March of 2015. This background section mostly describe 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 commonly deployed version of the mobile network architecture is depicted in Figure 2.1 and based on 3GPP TS 23.002 [3GP13g], with some minor nodes and network paths omitted[4]. The displayed architecture combines all three access technologies as well as the CS and the Packet Switching (PS) domains 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[5] the Serving GPRS Support Node (SGSN) and the Gateway GPRS Support Node (GGSN) are the core elements on the user traffic path. Elements other than these solve control plane tasks only.

One architectural detail to note is the strict separation between user plane and control plane tasks in the 3GPP architecture. Completely separate protocol stacks are used and signaling is mostly conducted in an explicit and out-of-band manner. This is contrary to the typical approach of the Internet's TCP/IP stack, especially its upper layers, where most state is implicitly inferred and only some signaled in-band.

The following sections give a short description of nodes and their tasks as well as used protocols stacks and signaling procedures for both the user and the control plane. The description will be mostly focused on the UMTS parts of the architecture which is also overviewed in [3GP12b].

2.1.1 *3GPP Radio Network*

The architecture has three 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[6] 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 protocols managing and

4 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]

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

6 Mobile devices are usually called Mobile Station (MS) in GSM networks and User Equipment (UE) in 3G and later. The terms are interchangeable and specifications often mix both terms.

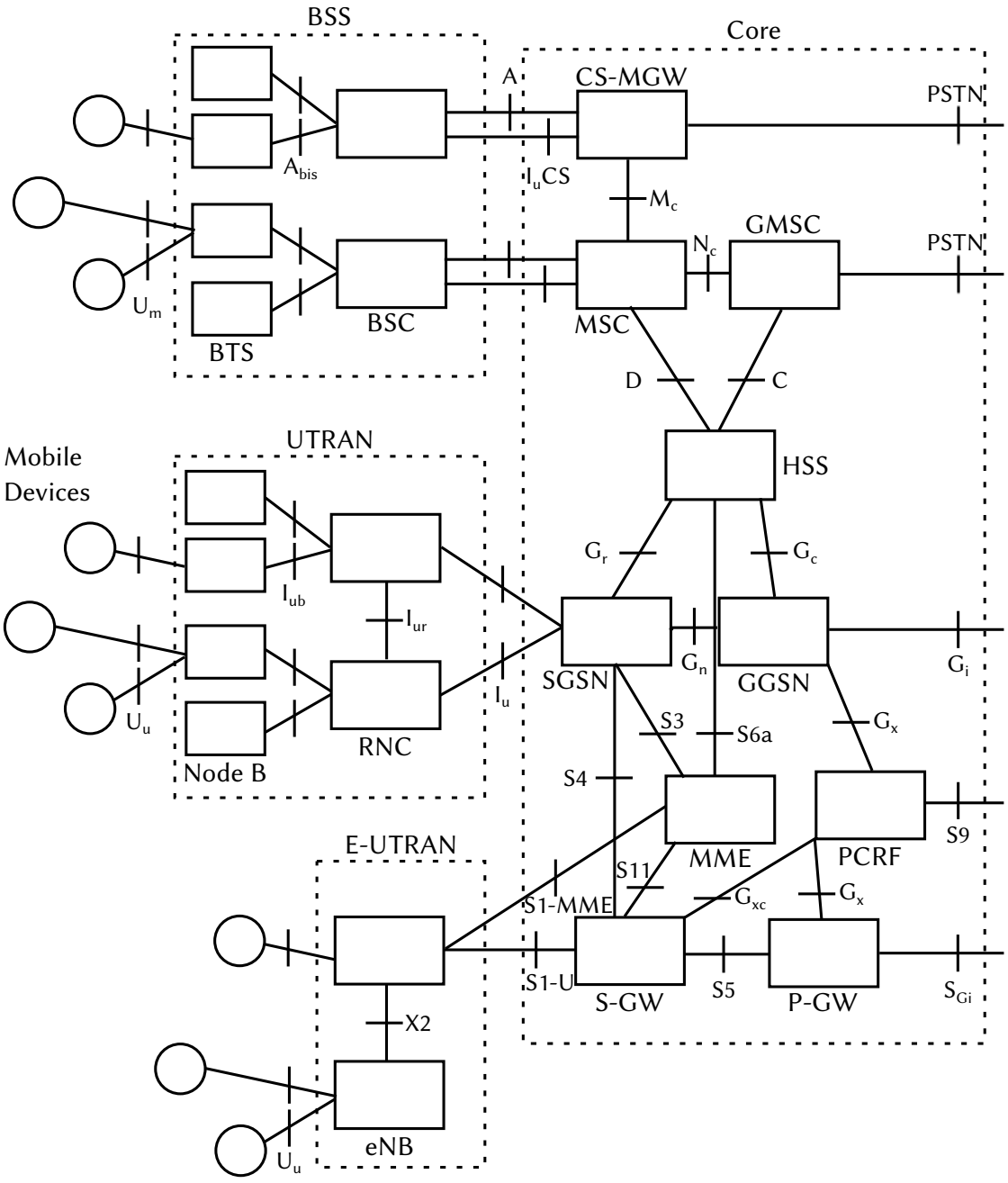


Figure 2.1: Overview of a combined CS/PS GSM/UMTS/LTE architecture.

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 Radio Resource Control (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 user plane 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 seamlessly handed over between cells without higher layers being informed or required to act upon such events.[7] The handover process is fully controlled and conducted by the network through 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 larger regions.

In UMTS multiple base stations are concentrated into one RNC. Most of the functions of a RNC, including Mobility Management (MM), are defined by the Radio Access Network Application Part (RANAP) control plane protocol defined in [3GP08h]. RANAP is used at the Iu interface between the RNC and the core network, i.e., the SGSN. Today, all non-radio links of the network are usually IP-based. But in the past all interfaces have also been explicitly defined for Asynchronous Transfer Mode (ATM) and exhibited some differences to their IP-based counterparts. 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.

2.1.2 *3GPP Core Network*

The PS domain of a mobile core network manages most of the aspects of the connected devices and acts as the gateway to the further general network infrastructure of the operator and 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. Each of the nodes can consist of any number of actual physical machines but are considered as a monolithic unit for the architecture and any related specification.

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 one core has to simultaneously provide support for both UMTS and LTE access, core nodes for both architectures have to be present. The exception are some EPC nodes that provide legacy interfaces to supplant their UMTS predecessors.

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

The two central elements in the user plane'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 GPRS Tunneling Protocol (GTP) based core tunnel, further transporting user plane traffic to the GGSN. GTP will be described in detail in Section 2.1.5. The GGSN's control plane tasks include mobility and connection management via RANAP to the RNC. 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.

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 every 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 version of the implemented infrastructure these two nodes can also be combined into one, eliminating the S5 interface and the GTP signaling between them. In the EPC many of the control plane tasks previously maintained by the SGSN have 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 [RFC6733] 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) which is integrated into the PGW. They act as a Deep Packet Inspection (DPI) entity, inspecting all user plane traffic and enable arbitrary filtering of traffic and traffic-based billing. Both entities are described in [3GP13i].

Figure 2.2 overviews the protocol stack on the path of the user traffic through the whole network from the UE to an external network.

2.1.3 *Core Network Concepts*

Most of the discussed upcoming research deals with the core network. Therefore, this next sections will explain in detail the concepts behind the 3G core network control plane and the GTP protocol family.

As mentioned, there is a strict separation between control instances and instances that carry the actual user traffic. Looking at the control plane side of things, management is conducted in a completely stateful way. Nodes keep track of every UE they are managing and need to locally store any state information they might need for management purposes. Of particular interest is the state revolving around the *Packet Data Protocol (PDP) Context*. For each open data connection a device has, both the GGSN and SGSN must keep such a PDP Context, which identifies the connection as well as the device belonging to it.

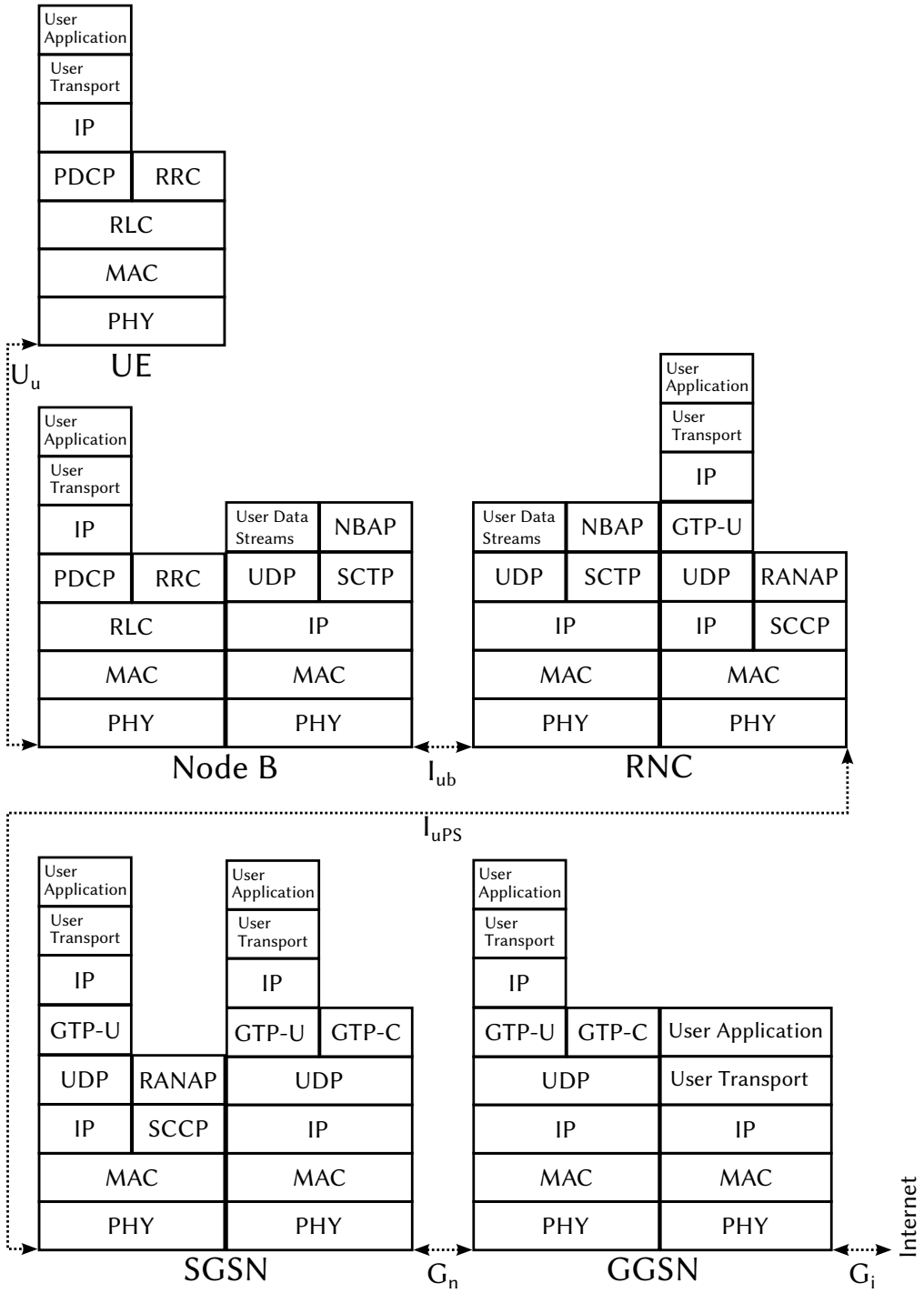


Figure 2.2: Simplified control plane and user plane IP-based protocol stacks on the user traffic path through the mobile network.

Additionally, a number of state machines are maintained. Transitions between machine states trigger a signaling message to specific network neighbors. Any one of these signaling interactions belongs to one or more larger control plane procedures. Most of the procedures that happen inside the core network PS domain or affect it are defined in TS 24.008 [3GP13f] and TS 23.060 [3GP13c].

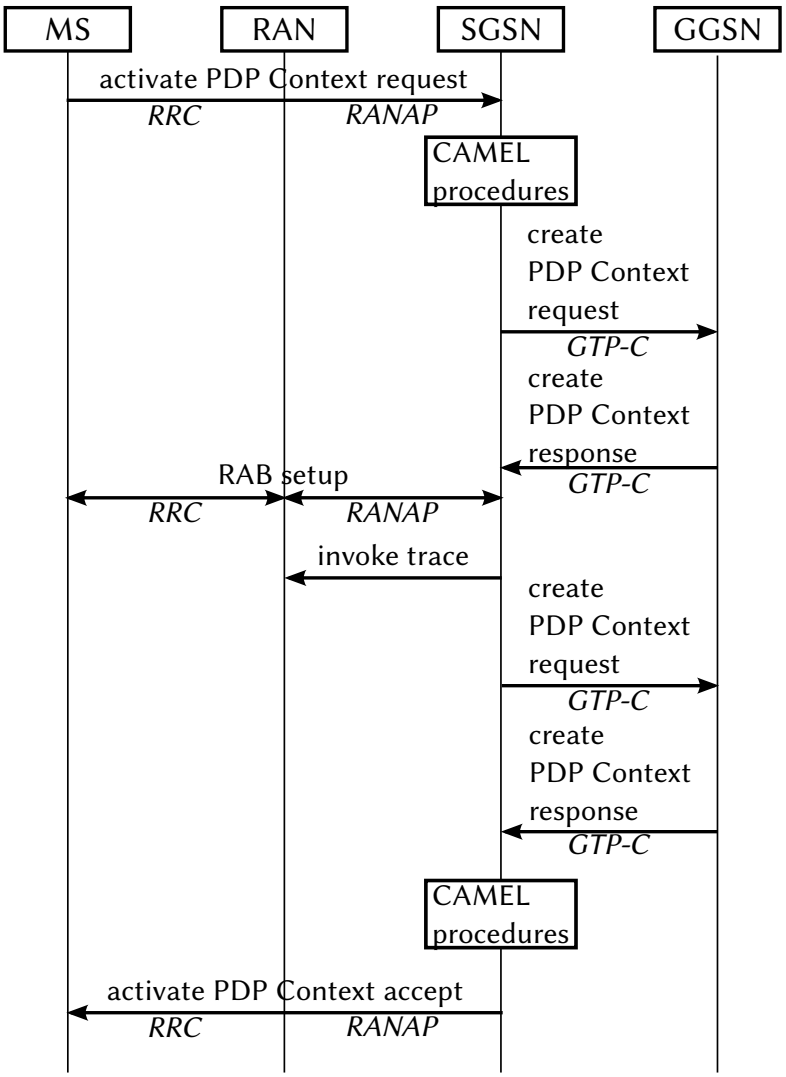


Figure 2.3: PDP Context activation procedure signaling interaction diagram for UMTS, including involved signaling protocols.

A very simple example is demonstrated in Figure 2.3. Initiated by the UE's session management state machine a data connection to the device is requested to be set up, triggering signaling with various protocols throughout the network. Additional secondary Customised Applications for

Mobile Networks Enhanced Logic (CAMEL) procedures, defined in [3GP07], are also triggered and conducted.

The general motif of control in 3G networks is rather different to that of the typical TCP/IP Internet stack. Whereas plain IP stacks rely on the end-to-end principle and put most of the control inside the devices at the edge, in 3G control procedures are spread across the network. Coordinating this requires the discussed control plane and all of the explicit signaling interactions. This results in a rather high complexity but also gives the opportunity to investigate these mechanisms and implications the structures have on the performance.

2.1.4 *Tunneling Concept: Bearers and PDP Contexts*

To enact the aforementioned user and control plane separation, a custom tunneling concept is used. Beginning at the UE, the actual user IP stack is never directly carried over the link's layer 1 and 2 protocols but always further encapsulated into so called *bearer*. The specifications distinguish between the *radio bearer*, existing solely on the air interface, the RAB, denominating the path between the UE and the CN, and the CN bearer between the SGSN and GGSN.

Technically, different protocols with tunneling capability are used on each link. PDCP is used on the radio link, GTP User (GTP-U) is employed between the Radio Access Network (RAN) and the CN and in the core between SGSN and GGSN. Closely related to the core network bearer are a number of information records stored and maintained at the two core nodes, the aforementioned PDP Context. For every active bearer a context is stored containing identification and management information about it. Amongst other data this includes device identifiers, e.g., International Mobile Subscriber Identity (IMSI) and International Mobile Equipment Identity (IMEI), tunnel identifiers (Tunnel Endpoint Identifier (TEID)), information about the intended public network through the Access Point Name (APN) field, charging, and QoS [3GP13c, Section 13].

Commonly, any device with an active data connection has at least one bearer and an associated PDP Context, the *default bearer*. In terms of QoS this represents a best effort tunnel carrying all user traffic that is not further differentiated. Additionally, the device can request a number of secondary tunnels, i.e., *dedicated bearers*, with certain QoS guarantees. Traffic matching a specified Traffic Flow Template (TFT) will then be carried over this dedicated bearer. However, this concept is scarcely used in 3G networks. All in all, one UE can be associated with up to eleven bearers, consisting of one default bearer and additional dedicated bearers.

2.1.5 *GTP and GTP-based Core Network Signaling*

A large part of core network communication is conducted by GTP. In 3G networks version 1 of the protocol, defined in TS 29.060 [3GP13b] and TS 29.281 [3GP13d][8], is used. For EPC some changes were made to GTP bringing it to version 2, which is specified in TS 29.274 [3GP13a].

8 For a more concise description of the protocol, one should actually read the GTP implementation in the community Free and open-source software (FOSS) project OpenGGSN at `https://github.com/osmobuntu/openggsn`.

The latter will not be further discussed as all presented evaluations will have a 3G network as basis.

GTP is mainly used on the Gn interface that connects the two main GPRS nodes, the GGSN and SGSN. Its functionality is split up between a user plane and a control plane part, named respectively GTP-U and GTP Control (GTP-C). GTP can best be described as an application layer signaling protocol and is intended to be transported by User Datagram Protocol (UDP).

	Bits							
Octets	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
1	Version			1	0	E	S	PN
2	Message Type							
3 **4**	Length							
5 **6** **7** **8**	Tunnel Endpoint Identifier							
9 **10**	Sequence Number							
11	N-PDU							
12	Next Extension Header Type							

Figure 2.4: General 12 B GTP header format.

Conceptually, GTP is structured through a base packet header, a number of extension headers and a message body consisting of a series of Information Element (IE). The base header is depicted in Figure 2.4 and has a total length of 12 B. Essential to the header are the TEID to identify the corresponding user plane tunnel and the 8 bit message type field. Each of the message types corresponds to a specific signaling interaction from the overarching control plane procedures. The procedures that GTP concerns itself with on the Gn path belong either to path management, tunnel management, or mobility management. Messages also usually come in request and response pairs, thus must be sent from the receiving node back to the original requester.

Each messages is defined as a specific set of IEs, each of which is either mandatory, conditional to some external factor, or optional. These IEs are of fixed or variable length depending on their type. They convey the actual state to be signaled and always relate to a specific UE and tunnel, for example the device's IMSI or the configured APN.

Coming back to the described PDP Context activation procedure of Figure 2.3, it contains both the GTP message *Create PDP Context request* as well as the *response* twice. Such a Create request

Table 2.1: All IEs in a Create PDP Context request, and respective sizes, for IPv4 network and user traffic only. The denoted sizes exclude the first message type byte.

IE	Presence	Size	IE	Presence	Size
IMSI	cond.	8 B	TFT	cond.	max 257 B
RAI	opt.	6 B	Trigger Id	opt.	var.
Recovery	opt.	1 B	OMC Identity	opt.	var.
Selection mode	cond.	1 B	Common Flags	opt.	3 B
TEID Data I	mand.	4 B	APN Restriction	opt.	3 B
TEID Control Plane	cond.	4 B	RAT	opt.	3 B
NSAPI	mand.	1 B	User Location Information	opt.	10 B
Linked NSAPI	cond.	1 B	MS Time Zone	opt.	4 B
Charging Characteristics	cond.	2 B	IMEI (SV)	cond.	10 B
Trace Reference	opt.	2 B	CAMEL Charging Information Container	opt.	var.
Trace Type	opt.	2 B	Additional Trace Info	opt.	11 B
End User Address	cond.	8 B	Correlation-ID	opt.	3 B
APN	cond.	max 102 B	Evolved Allocation Retention Priority I	opt.	3 B
PCO	opt.	max 255 B	Extended Common Flags	opt.	3 B
SGSN signaling address	mand.	6 B	User CSG Information	opt.	10 B
SGSN user traffic address	mand.	6 B	APN-AMBR	opt.	11 B
MSISDN	cond.	max 17 B	Signaling Priority Indication	opt.	3 B
QoS Profile	mand.	max 257 B	Private Extension	opt.	var.

consists of the 36 IE depicted in Table 2.1. Neglecting the elements which have no predefined upper length bound (besides the default 16 bit IE length field) and assuming a maximum length for the other variable elements this results in a message size of 1059 B. The complexity of the other message types is comparable.

One of the investigations that will be conducted is that of the core network load, which will be defined and discussed later in detail. GTP messages could play an interesting role here as they may directly or indirectly contribute to this load or at least be an indicator of load existing otherwise. Load could be caused by the generated network traffic as well as the assembly, processing, and storage of the involved state in form of the IE.

The following sections detail the three GTP tunnel management message pairs involved in the maintenance of PDP Contexts. These are the *Create, Update,* and *Delete PDP Context requests* and *responses.* They represent the basis for the core network investigations.

2.1.5.1 *Create PDP Context Message*

This message type is part of procedures that enable the PS data connection and the GTP tunnel for a mobile device. These are the **PDP Context activation procedure** (already depicted in Figure 2.3) and the **Secondary PDP Context activation** for additional GTP tunnels to the device with specific QoS levels set. They are triggered by the mobile device through RRC/RANAP signaling during or following a GPRS attach procedure.

When a GGSN receives a create request from an SGSN, it has to allocate the necessary resources for a PDP Context. Depending on the outcome, a response is sent back, indicating the success or failure of the operation. Typical failures include failed user authentication, a lack of resource, or unrecoverable system failures and malformed or corrupted request.

2.1.5.2 *Delete PDP Context Message*

Similar to Create Context messages, a **Delete PDP Context request** and **response** always coincides with the termination of a GTP tunnel and the removal of the associated PDP Context. Create and Delete requests together mark the beginning and the end of every user traffic tunnel, making them very interesting in determining tunnel properties and perfect candidates to indirectly identify tunnel durations in a core network investigation.

Deletes are either created through explicit PDP Context deactivation procedures or play a part in GPRS attach and detach procedures. Contrary to creates, they can be initiated from the MS as well as from a core network node, depending on the kind of procedure.

2.1.5.3 *Update PDP Context Messages*

Several procedures also emit **Update PDP Context requests** and **responses**, usually in relation to some aspect of the tunnel or the device changing. Possible causes for an *Update Context request* are:

- 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 *PDP Context Modification* procedure.
- 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.

However, the appearance of update message signaling in some of these procedures is conditional or even optional. This often depends on the specific implementation and is not known without in-depth knowledge of it. Only for mobility management procedures the updates are mandatory.

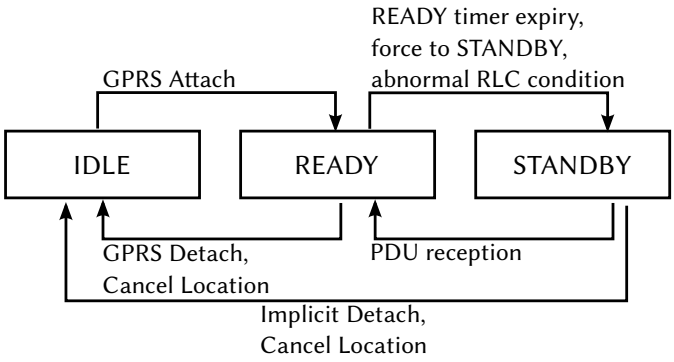
By observing update messages one could capture most forms of mobility happening in the network, and get a good picture of potential correlation between mobility and tunneling characteristics. By distinguishing portions of tunnels which were associated with a UMTS RAB from Second Generation (2G) radio access through the related update message, one could also study any influence of the access technology on the core.

Nowadays GSM/GPRS is either used in older models or feature phones or in mobile scenarios in rural areas where GSM still is prevalent due to its usage of lower frequency bands and thus larger coverage. Both could indicate that the data session will be rather short because of limited device capabilities or low throughput rates of GPRS.

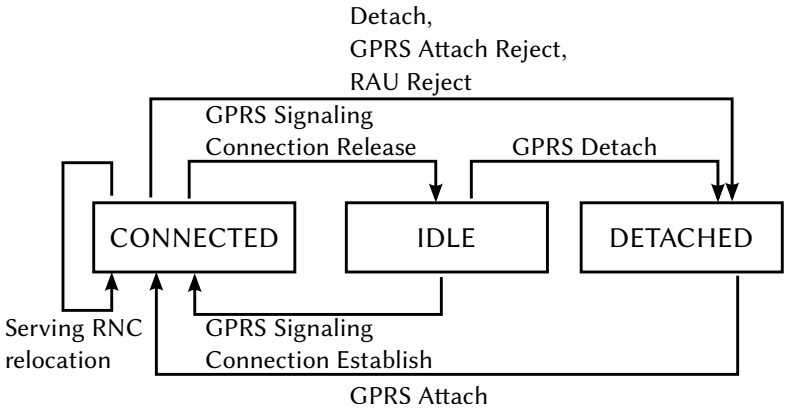
2.1.6 *GTP Influencing State Machines*

To understand the occurrence of these signaling procedures one should look at the state machines that govern these. Involved in the tunnel management aspects are three distinct finite-state machines (FSMs), namely the MM and RRC state machines and the actual PDP state model. The MM and RRC describe the current state of the mobile device and its radio and data connection. They are both maintained at the MS and mirrored at the SGSN.

The MM model, defined in [3GP13c, Section 6.1], describes the general state of the data connection. State switches occur either based on an idle timer or when new packets arrive for the mobile device. The specific model depends on the currently used RAT with only slight differences between GSM (Figure 2.5a) and UMTS (Figure 2.5b) access. The LTE related model brings larger changes but is omitted here, as it will not be relevant for the investigation. With the transition to and from the **IDLE** state in the 2G model (or **DETACHED** in 3G) **GPRS Attach/Detach** procedures are triggered, also resulting in the transmission of PDP Create and



(a) State machine for 2G radio access.



(b) State machine for 3G radio access.

Figure 2.5: SGSN MM state models and machines as defined in [3GP13c, Section 6.1].

Delete Context messages. Likewise, other state transition procedures indicate mobility and location changes, which usually include update messages.

The RRC state machine given in TS 25.331 [3GP12f, Section 7.1] and depicted in Figure 2.6 governs the usage of radio channels and therefore power states of the MS. State changes happen depending on user and radio activity and inactivity which is determined by timers. Only in the Dedicated Channel (CELL_DCH) state is the MS assigned a dedicated channel for its data connection and can transmit at full bidirectionally. But this consumes the most device power and radio resources, both of which are scarce. The goal of the state machine is to minimize resource usage with intermediary states — Forward Access Channel (CELL_FACH), URA Paging Channel (URA_PCH), and Cell Paging Channel (CELL_PCH) — that successively require less power and radio channels, before completely turning of the RRC connection by transitioning to the idle state. Coinciding with the RRC, the CN GTP tunnel can also be released or needs to be reestablished. However, this is implementation specific and not precisely specified.

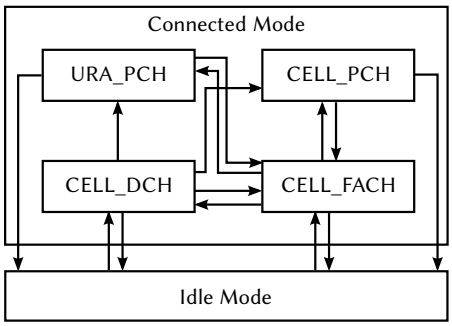


Figure 2.6: RRC State Model as per [3GP12f, Section 7.1].

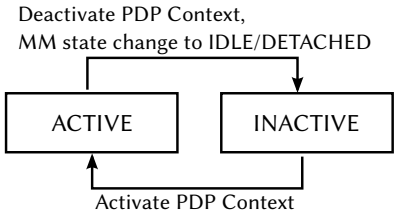


Figure 2.7: PDP State Model defined in [3GP13c, Section 9].

The final state machine of relevance is the **PDP State Model** from TS 23.060 [3GP13c, Section 9] in Figure 2.7. It reflects the actual state of the PDP context and associated tunnel and is synchronized with the MM state machine.

2.1.7 *Signaling Discussion*

This section on the basics of current mobile network architectures serves a critical purpose: In order to measure and evaluate network traffic one has to first understand its architecture and needs to grasp how certain traffic patterns can occur. Unfortunately, the 3GPP specifications do not make this task very easy. Typical IETF protocols and architectures adhere to the fundamental principles of protocol layering, function separation and end-to-end. One can read a single Request for Comments (RFC) and understand its function and directly implement it independent of the knowledge of any other specification. This is not the case in a 3G network. Functions are often spread over several protocols or nodes, necessary details essential to an implementation are spread out over several specifications without direct reference or are even completely omitted. This circumstance makes it very hard to attribute certain observed phenomena to a specific feature in the specification.

The protocols used in 3G networks are also very heavy in terms of state and signaling inside the network. This can be the cause of unintended and hard-to-predict load, which will be defined and discussed in Section 2.3. For now, some initial load information in relation to the Create, Update and Delete PDP Context Request and Reply message pairs can already be deduced. Measuring the time delta between corresponding Create and Delete events obviously results in

the total duration a tunnel was established. Having shorter tunnels often also means having a greater number of tunnels and therefore a higher volume of signaling messages and an increase in processing and state-keeping efforts due to the signaling.

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. 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. 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.

2.2 Related Work

The investigations conducted in both this and the subsequent chapter do not fall strictly into an existing research category but instead aim to provide diverse insights into the control plane from the perspective of the core network. Nonetheless, a selection of publications from the tackled fields is collected here and the interesting aspects for this work are noted. In the following sections the related work is divided into four distinct fields.

Work in the first and second sections evaluate properties of the mobile network and its traffic. They are distinguished in their approach to the investigation, as the first group uses active measurements from mobile devices or conclude from other sources of traffic whereas to the other one has access to passive measurements from inside a 3G mobile network. Publications from the third category can be generally subsumed under the term *traffic modeling* and may not be specific to cellular networks. The final field concerns itself with investigative work conducted by the responsible standardization and organizational bodies themselves, i.e., the 3GPP and GSM Association (GSMA).

2.2.1 *Device Active Measurement Investigations*

The approach taken by active measurement studies is simple yet still very insightful. They are performed by writing custom application layer measurement programs for a mobile device. Specific traffic patterns are then generated, recorded, and evaluated. While this can provide very detailed information about the higher network layers, it is limited both in lower layer information as well as scale, due to being limited to a rather low number of devices.

Despite being more or less completely specified in the 3GPP documents, there is no open layer 1 and 2 (together also called "baseband") implementation for 3G.[9] Therefore, the baseband's

9 Apart from OsmocomBB (`http://bb.osmocom.org/trac/`), but it only provides GSM and partial GPRS functionality.

behavior can not be directly instrumented from the application layer. Attempts to infer some properties are still worth conducting as the following selection of publication demonstrates.

In [Xu+11] Xu et al. use data from a location service combined with active measurements to determine the possible geographic location of a GGSN in order to improve the location of application content caches for the current network infrastructure. Similarly, in [Wan+11] Wang et al. developed a program to probe mobile networks for middle boxes. That term includes any node that alters traffic and affects performance not intended by the actual end-to-end protocols. Examples are Carrier-grade NAT (CGN) [RFC7021], firewalls, or intercepting HyperText Transfer Protocol (HTTP) proxies. A large number of such nodes were present in the investigated mobile networks and resulted in increased device power usage and download durations and even pose security issues themselves.

Concerning methods to infer specific baseband and RRC state machine timer values with active measurements, a 2007 paper [BRB08] presents a way to do this by transmitting packets with a varying inter-departure time and studying the resulting arrival pattern. Indeed, the dynamics of the radio interface's RRC signaling and involved state machines are under investigation by several publications. However, almost all focus solely on the impact at the radio interface but pay little attention to potential implications in the CN.

The aforementioned work is continued in [Per+09] and uses the presented tools to derive RRC transitions and power usage from traffic patterns. They found, that operators have a rather larger freedom in configuring the mobile network control plane state machines and deviate from the standard and even omit some states completely.

A further example of cross-layer influences in mobile cellular networks is [Qia+11]. It discusses the impact of application layer behavior on RRC signaling 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.

This is further elaborated on by research from Schwartz et al.[Sch+13] using the same technique to analyze the radio signaling load and thus power efficiency from several mobile phone applications. The impact of custom set state machine timers interacting with application traffic is further investigated and the QoE is investigated.

2.2.2 *Research Based On Network Traces*

The second approach to mobile network investigations comes in the form of recording and evaluation traffic traces inside the network. This brings a much larger experiment scale with it, albeit usually at the cost of some finer grained details in the higher protocol layers because of aggregation to flow level. With core network measurements, the signaling traffic of the observed link can also be directly investigated, which is a huge benefit compared to the guesswork in active measurements.

The authors of [RRCpt] investigate the influence of individual CN nodes on the one-way delay distribution of user traffic packets. According to the work, the latency portion added by the SGSN is larger but also fluctuating more, while the GGSN added a small but steady amount of

latency. This provides initial clues on the expected load impact of the CN for the investigations in this work.

Following up on the topic of mobile network one-way delays is Laner et al. in [Lan+12a]. The end-to-end latency of an early LTE/EPC network implementation is compared to that of a HSPA network at several measurement points in the networks. The results show a lower median latency for LTE, despite some scenarios still being in favor of 3G networks.

The authors of [Sha+12] limit their focus to a specific subset of connected devices, namely those of Machine-to-Machine (M2M) type. These are small automated devices that periodically send out data, e.g., sensor readings, or receive control commands. The paper attempts to characterize these on the basis of their generated mobile network traffic. The patterns are clearly distinguishable from traffic caused by other device types such as smartphones.

A 2012 publication [ZÅ12] presents a more general look on the traffic composition of cellular access networks in comparison to a wired access network. More and shorter flows are occurring in the case of cellular networks. It will be interesting to see if this shorter-but-more theme is also evident in signaling traffic. Additionally, even traffic pattern distinctions between types of applications are made showing a wide range of possible outcomes across the investigated applications.

Both the authors of [Sha+11] and [Pau+11] 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. Additionally, [Bae+11] delivers a theoretical introduction on how to conduct large scale network measurements and compares some data evaluation approaches. The 2008 paper of [RHR08] takes a look at times scales and time of day deviations observed in aggregated user traffic in a mobile network.

Up until now no trace-based investigation considered the control plane in their evaluation. The following publications include this at least to some degree.

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 paper [LBW07] combines simulations based on WiFi and synthetic traces with prior knowledge of RRC states and their effects to investigate detection methods for signaling DDoS occurring on the radio interface. A possible magnitude of this type of attack is discussed. This also gives an indication of the correlation between user traffic patterns and radio signaling.

A 2010 publication[Qia+10] 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 the device's energy consumption for the traffic patterns seen in the data.

A more recent publication at [He+12] performs a RRC investigation at the path between RNC and SGSN. 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 final paper [RCD10] recaps some general attack scenarios on 3G networks that exploit the specific 3GPP system design. These are often closely related to the control plane.

2.2.3 *Traffic Modeling*

Extracting viable models from mobile traffic measurements will also play a significant role onwards. The first related work is a survey of source modeling approaches for GPRS user traffic from the year 2000 [SLT00]. Models for HTTP traffic and user behavior are compared and a combined model is recommended. One has to keep in mind, though, that due to the rapid developments in the Web in recent years those models might no longer be valid.

Similarly, the authors of [KLL01] derive a synthetic UMTS traffic model from wired dial-up traces. By using a batch Markovian arrival process they characterize session traffic in most cases with a lognormal distribution.

Work conducted in [HW05] derives a model for the users' mobility in a mobile network. The mobility model is however more focused on the circuit-switched voice communication features of a phone. Likewise, the authors of [PPF05] introduce a traffic model for Session Initiation Protocol (SIP) Voice over IP (VoIP) communication in UMTS networks. However, this model is specific to the IP Multimedia Subsystem (IMS) domain of UMTS and potentially not applicable to the more common over-the-top pure SIP traffic. The model additionally investigates some initial UMTS control plane timing values, such as the processing time of PDP context activation messages.

A further publication in 2005 [LK05] attempts to model the delay of IP packets passing through an UMTS network using a batch Markovian arrival process. However, the model specifically focuses solely on the delay originating from processing at the radio link and not at the core nodes.

Finally, a further paper by Laner et al. [Lan+12b] investigates, amongst other things, a user's session duration and throughput in a HSDPA network. The duration is modeled as an exponential distributions and the throughput using a lognormal distribution, albeit both exhibit additional heavy tail characteristics.

2.2.4 *3GPP and GSMA Related Work*

The two associations related to the mobile network under scrutiny, the 3GPP as well as the GSMA themselves have also released some studies and recommendations concerning potential effects of and issues with the control plane.

In reaction to the mentioned RRC signaling DDoS the GSMA released some best practices [GSM12] intended to reduce the number of signaling messages in these circumstances. The cause of the DDoS were in most cases mobile devices that circumvented the RRC state transition timers and explicitly switched the radio to the idle state after a transmission was finished re-enabling it whenever needed. This can greatly reduce the power usage but increases the

number of signaling messages to be sent and thus the load in the radio network and possibly also inside the CN. With the presented "Fast Dormancy" mechanism, mobile devices are supposed to reduce the radio signaling amount while still saving energy. The implications of this mechanism on the core are not investigated.

A 3GPP-released study [3GP11] also describes the diverse traffic mix originating from modern smartphones and its associated signaling problems.

The aim of the study in TS 23.843 [3GP13j] is to document some of the control plane bottlenecks and attack vectors on the CN. This also includes the interesting case of GTP-C overload and causes for this scenario. Some approaches to alleviate the effects are also presented but mostly targeted at the EPC. The final study is an extension to the last one [3GP13k] and focuses solely on GTP-C overload control to be included in a future version of the 3GPP architecture. Therefore, the mostly unfinished document again targets just a future version of LTE and provides no investigation of the actual load situation in current 3G networks.

All of the presented publications relate only to some degree to the forthcoming investigations. The combination of the aspects of CN signaling with a statistical evaluation and load modeling of PDP contexts should be a genuine contribution of the thesis.

2.3 Mobile Core Network Load

Now that both the basic architecture and protocols are and introduced and related work is presented, a specific perspective on the CN control plane can be defined. Existing core network measurement studies looked at the control plane mostly in a rather incoherent manner. Some aspects were singled out and presented without forming an overarching motif. The driving question for this research aspect was that of core network load. This section defines load metrics in this context. Following afterwards is a discussion on potential factors that could influence this load.

2.3.1 *Load Definition*

A traditional definition of link load or utilization ρ_l is the ratio of the used versus the available bandwidth on a link

$$\rho_l = \frac{b_u}{b_a}. \tag{2.1}$$

The network load ρ_N can then simply be defined as the average load of all involved links,

$$\rho_N = \frac{\sum_{i \in N} \rho_{l,i}}{|N|}. \tag{2.2}$$

However, the link itself is not the only component that has a limited capacity and thus can experience load. Other load metrics might be defined using different limited resources as well, e.g., the available memory or processing power.

In Internet core routers those other factors are mostly well known and researched. Their main functionality is to forward packets on the basis of a routing table. This table is generated by exterior and interior gateway protocols, usually Border Gateway Protocol (BGP) in conjunction with Routing Information Protocol (RIP) or Open Shortest Path First (OSPF), and may grow rather large[10]. The generation and maintenance of a routing table including all lookups — which might be expensive in a large table — incurs load on the router's available memory and processing capacity. This kind of load can be attributed to the router's control plane and occurs in addition to the user plane packet forwarding load. All in all, the Internet's control plane was designed to be lightweight and isolated. Only minimal and distributed state is kept where necessary.

The situation is a bit different in a mobile network. Here, the control and user plane are tightly coupled as discussed in Section 2.1. Therefore, load of individual resources cannot be looked at separately anymore and a node will be limited by any one of these resources. The load of any particular mobile core network node ρ_n could then be defined as the maximum of the node's link load $\rho_{l,n}$, memory load $\rho_{m,n}$, and processing load $\rho_{p,n}$,

$$\rho_n = \max(\rho_{l,n}, \rho_{m,n}, \rho_{p,n}). \tag{2.3}$$

The CN load ρ_{CN} is then the maximum load of any of the CN nodes,

$$\rho_{CN} = \max_{n \in CN}(\rho_n). \tag{2.4}$$

In this definition the performance of the CN can limited by any one of the involved core nodes. Looking back at the discussion of the 3G architecture this should be an appropriate definition.

But before attempting to apply the model to an actual mobile network one additional limitation that reflects the situation in real mobile networks has to be introduced first. Core network nodes should be considered as black boxes. They are custom pieces of hardware sold by vendors as-is, providing no opportunity to directly monitor the inner workings of the node, including memory and processor usage. Only the network traffic leaving these nodes can be directly tapped and investigated as will be described in Section 2.4. But the amount of signaling traffic exchanged on these observable links could give ample opportunities to indirectly infer some of the nodes' inner state and load.

With the basics of the architecture in mind, the GGSN can be considered a top candidate for being a load bottleneck. All traffic leaving or entering the packet switched domain must pass through this element, and it is involved in all CN 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, the GGSN will be the node under scrutiny in the trace evaluation and performance model creation.

10 Compare, e.g., the number of BGP entries given in http://www.cidr-report.org/as2.0/.

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 acts as the interface to the radio network as well, which involves handling RABs and mobility management. However, it can be assumed that the number of SGSNs employed in a mobile network is larger than that of GGSNs, as they are typically kept closer to the regionally distributed radio networks. This means that a single node would have to handle less mobile devices and related signaling interactions. 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 be very load-sensitive as well. 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. Hence, this investigation concentrates just on the case of the GGSN.

2.3.2 *Load Influencing Factors*

With the described understanding of core network load at hand, one can now speculate on factors that could influence the load in mobile networks. Such factors will also play an important role for the following evaluation.

The first factor comprises the mobile devices themselves. They are the source of any user traffic and the cause for most signaling procedures, for some procedures directly but for most indirectly. But it stands to reason that the device factor is only an aggregate of several influence subfactors. And the specific selection of subfactors and their parametrization will be unique to each device.

Specifically, this factor includes the type of device — which in turn is a composite of the hardware, the baseband as well as the OS — and the running applications. The usage of applications decides when the device should establish a mobile data connection, how long the connection is held, or which mobile technology takes preference. Depending on the RAT in use, subtle behavioral and signaling differences can be expected, e.g., in the timing of the radio transmission intervals, which could influence the investigation. Some specific GTP 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 influencing the duration of transport layer connections and therefore also the underlying tunnels. Some further influence factors of the protocol stack are discussed in Section 5.1.

The actual user-traffic patterns are also generated by the applications running on the device. The application traffic spectrum ranges from low volume but extremely long duration instant messaging applications over recurring ad-retrieval patterns up to short duration burst downloads. Since the mobile application ecosystem is very versatile every device will pose a unique combination of applications. The governing factor in everything device-related is the user and her behavioral patterns. This expresses itself both in the traffic dynamics and in the mobility

pattern. But it is nigh impossible to single out individual behavior in a network's traffic mix or a large network trace.

Easier to observe are the temporal statistics of large user groups, not targeting individual users but the overall effects of device usage in a certain time span, e.g., based on the time of day or the day of the week. 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. Studies investigating this typically only look at user traffic. It should prove interesting to find out if the CN control plane shows similar patterns and can be correlated to user traffic.

The second large influence factor are the mobile network's control plane state machines and their related signaling procedures. If a network-side state machine inactivity timer decides to remove an existing tunnel, signaling will occur, which suggests there will be a large number of tunnels with a duration in this range. While most 3G control plane timers have default values, they are often changed by the manufacturer or network operator and will vary from one implementation to another. It is therefore quite difficult to give any hard numbers in advance, one has to correlate such aspects with certain events in the results.

2.4 Evaluation Methodology

With the mobile network load defined and possible influencing factors described, the findings can now be applied to an actual mobile network. For this data from passive network traces will be employed. Before that, this section describes the monitoring setup and the captured data. This also includes a description of some methods required to examine specific device types and other device-based factors from the dataset.

While this chapter only employs passive measurements, Chapter 6 will additionally deal with approaches to conduct meaningful active device-based measurements and set up a mobile streaming simulation testbed based on some of the results.

2.4.1 *Network and Monitoring Setup*

For the analysis the METAWIN monitoring system, developed in a previous third-party research project and deployed in the network of an Austrian mobile operator, is used. Detail information on this setup can be found in [Ric+06].

The measurement taps are located at the Gn interface at one GGSN within the core network as depicted in Figure 2.8. It gives access to a wide spectrum of core GTP signaling, including the mobility and tunnel management. The system does not offer a complete packet trace, but aggregates every signaling transaction and user traffic flow down to a number of select fields. This includes GTP IEs such as the RAT as well as the terminal types of the mobile clients. The latter is determinable by the TAC-part of the IMEI and will be discussed later in detail.

In the network under study, a direct link between GGSNs and the RNCs and circumventing the SGSN is present. It is only used for transporting user-plane traffic under specific circumstances, and signaling procedures are still carried out in the normal way between SGSNs and GGSN.

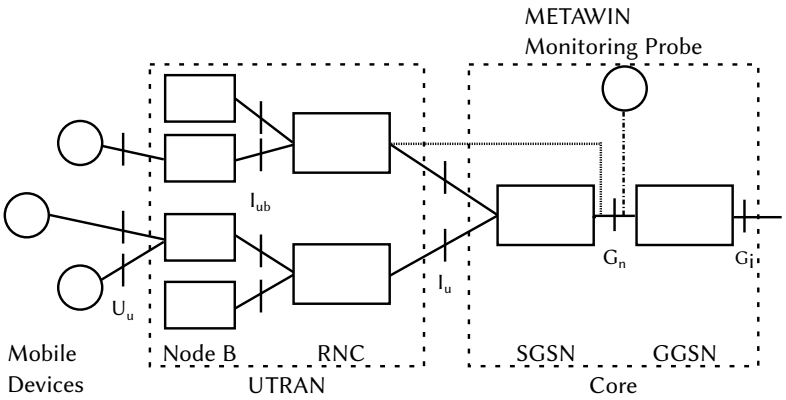


Figure 2.8: Location of the METAWIN monitoring probe in the 3G core network.

Therefore, only the Gn interface at GGSN is seeing the complete core network traffic, which explains the location of the tap. The network under study has more than one GGSN at different physical locations. The tapped GGSN manages about half of the operator's total traffic volume in this period.

Recording data in a live network necessitates meeting strict privacy requirements regarding the handling of user-related data. METAWIN complies with this by anonymizing all user-identifying markers. Application-level payload is removed and all remaining user-specific data (e.g. the IMSI) are non-reversibly hashed before recording. UEs in a dataset can still be differentiated by the hashes but not traced back to the actual user. The wiretaps deployed within the monitoring system are time-synchronized with Global Positioning System (GPS). Accordingly, the packet timestamps have an accuracy of least 100 ns.

2.4.2 *Dataset Description*

Using METAWIN a week-long core trace was acquired. It was recorded in April 2011, specifically beginning at Monday, 2011-04-10, 00:00:00 and ending Sunday, 2011-04-17, 23:59:59.

The trace includes user plane as well as control plane traffic. User plane traffic is recorded at a traffic flow granularity level with the trace containing data on 2.2×10^9 aggregated flows. No exact flow start time is given, instead it is rounded down to a 2 h window, with the timestamp just marking the beginning of the window. A flow entry further consists of hashed identifiers for the IMSI and the remote server. Besides the usual protocol and port information, the transmitted data volume, in a number of packet as well as byte count, is given on in both link directions. Additional extended information is stored on HTTP traffic. This portion of the trace includes precise timestamps as well as the MIME-type, result code, and size of the requested objected.

The recorded control plane traffic consists of 4.1×10^8 GTP tunnel management transactions, i.e., every Create, Update, and Delete request and response. Not all of the IEs' data is included. But most importantly, it includes the Type Allocation Code (TAC), RAT and hashed IMSI for the purpose of device discrimination. Also present are several timestamps with 64 bit precision

describing the time of the request, response and the tunnel's start time. Finally, the GTP data contains the response codes for each request. With these codes, failed transactions can be distinguished from successful ones and examined separately. Since the hashed IMEI is consistent across the user and control plane data, both can be cross-correlated.

All trace information was exported from METAWIN as pure line-based text data. For this investigation all records were fed into a SQL database. Evaluations were then conducted through scripted queries on the database using Python scripts and further statistically evaluated in R.

2.4.3 *Device Identification*

Individual device types in a mobile network can be identified in the data through the TAC field on every entry. The TAC, defined in [3GP13h], represents the first eight decimal digits of the IMEI and uniquely identifies each device type. The following six digits of the IMEI constitute the serial number of a specific device, which is omitted in the data to protect the privacy of subscribers. Due to the short length of this serial number, popular devices will often be assigned more than one TAC, somewhat complicating the identification of certain device models.

TACs are assigned to individual device models by the regional members, or Reporting Body Identifiers (RBIs), of the GSMA, distinguished by the first two digits of the TAC. The full allocation information is not freely available, but only to members of the GSMA, which is not a viable option for research institutions and other interested parties. Some independent efforts have been made to collect TACs from devices. Most of them allow just low-volume queries for specific TACs for non-commercial purposes. However, one TAC dataset is publicly available and can be used freely.[11]

This evaluation uses this dataset with some additional device identifiers and classification annotations collected during the course of the investigation. With this at hand, most of the devices associated with the flows and GTP messages from the trace were identified and categorized.

2.4.4 *TAC Evaluation Validity*

It is important to know whether the information available in the TAC dataset covers enough of the devices seen in the traces to conduct sufficiently meaningful evaluations. After all, the TAC data is large but might still be very incomplete due to the sheer number of devices in existence.

Table 2.2 provides statistics on the devices that could be identified in the trace. About 81 % of all unique TACs present in the trace could be mapped to a known device. More importantly, when looking at the total number of tunnels and GTP messages during the week, even 91 % of the responsible device can be determined. Finally, the flow data shows an even clearer picture, as almost all of the devices involved can be identified.

This is an interesting result in itself, as the 19 % of devices present in the dataset that could not be identified through the TAC are the cause for only about 9 % of signaling and 0.3 % of

11 Available at: http://www.mulliner.org/tacdb/.

Table 2.2: Relative TAC statistics.

Type	Relative number of devices with an entry in the TAC dataset
Total number of flows	99.72 %
Ratio of total traffic	99.97 %
Total number of tunnels	87.57 %
Total number of GTP signaling messages	90.95 %
Number of distinct UEs	80.93 %

total traffic. This means there is a long tail of device types in this mobile network with very little impact on load-influencing factors. With these results, one can be rather confident that evaluations using device discrimination based on this TAC mapping should give viable results.

2.4.5 *Device Classification*

With these device-to-TACs mappings available, additional meta-information can now be added to it, intended to distinguish some of the described load influencing factors. Knowing the model gives also a good knowledge of the device's category and of the OS it is running by default.[12]

The device's category represents a general classification of the device and should give some initial hints on the fields of use. The devices are partitioned into smartphones, feature phones, 3G USB dongles or 3G+WiFi routers, and all other devices. The term "feature phone" usually points to low-end mobile phones with at least some kind of data capability, often with a physical numerical keyboard. Phones that could subjectively fall into either the smartphone or the feature phone category were generally attributed as smartphone. Not covered here are any kind of M2M devices, because the TAC mappings are very inconclusive and incomplete in this area. And concluding from the previous Table 2.2, the impact of such M2M devices should be negligible.

The next classification variable is the OS. Most popular in the trace were the two dominant smartphone OSs, Android and iOS, but also Symbian[13], often found on feature phones, was present. Other systems of note are Blackberry OS and Windows Phone or Windows Mobile, but they occur in such a low volume in the trace that it was decided to completely neglect them and count them towards the other and unknown devices. It should also be noted that USB dongles and routers cannot be linked to any specific OS solely by the knowledge of the TAC. Also not

12 The OS actually running on the device at the time of the measurement can not be inferred on this way. But the number of devices running a different OS than the one installed by default should be relatively low.

13 While not completely accurate phones running Series 40 were also attributed to this category because of their close relationship.

distinguishable are the exact release versions of the OS on a specific device. This could diminish the evaluations, as the network behavior could change noticeably between two major versions.

With this knowledge, one can even conjecture about the applications running on the device. Combining the OS with lists of the most popular applications for this platform can already give some very helpful hints on what can be expected from the traffic mix these types of devices are generating. One final possible TAC classification could be a categorization by the phone vendor. However, this was not conducted because it can be safely assumed that the information gain is negligible in comparison to the device type and OS, e.g., iOS will be installed on every Apple smartphone anyway.

2.4.6 *Preliminary Device Statistics*

After applying the categorization to the network dataset the device composition is evaluated to get a first grasp of the network's makeup and to help understand the later investigations.

Smartphones and 3G dongles form the two largest observed portions of shares of devices, while classic feature phones do not seem to play a major role anymore. About twice as many Android as iOS devices are present, attributable either to the contractual situation of the operator or the wider price range of Android devices.

Regarding traffic, feature phones generate negligible amounts of user traffic despite still making up one tenth of the device fraction. 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.

One observation across all device types is that about 14 % of all mobile devices have activated their GPRS data service and GTP tunnel and cause signaling traffic, but do not initiate any user plane traffic at all. It is unclear if this is an intended behavior as this will lead to an increase of the devices' power usage and of radio spectrum resources with seemingly no benefit to the user.

2.4.7 *Statistical Methods*

As a final preparation for the evaluation all the statistical tools that will be used in the evaluation, are briefly defined in this section with material based on [FMF12] and [Knu97].

2.4.7.1 *Distribution Functions and Fitting*

With a distribution function, also called Cumulative Distribution Function (CDF), a monotonous mapping of continuous values to a probability can be well represented. It is defined as the probability that a random variable X is less than or equal to a value x,

$$F(x) = P(X \leq x). \tag{2.5}$$

Sample of real data are generally finite and not continuous. Hence, the distribution can only be approximated by an Empirical CDF (ECDF) $F_n(x)$ for values $X_1, X_2, \dots, X_n$ and

$$F_n(x) = \frac{\text{number of } X_1, X_2, \dots, X_i \leq x}{n}. \tag{2.6}$$

One of the analysis's goal is to break down the actual measured system to a simplified probability model. This can be conducted by attempting to match the empirical data distribution to an existing basic probability distribution, e.g., exponential, Gamma, log-normal, or Weibull. In order to achieve this one of several readily available matching methods, which rely either on closed formulas or numerical optimization, can be used. Two simple methods are *Matching Moments* [Vos00, pp. 99-143] and *Maximum Likelihood*.

The former estimates parameters for a preselected distribution function by optimizing the target distribution function so that its moments converge to those of the sample data. The latter approach finds a fitting target probability function by calculating the log-likelihood of the data for a preselected distribution and maximizing the likelihood.

In such cases where none of the basic probability distributions proved to be a good fit an attempt was made to converge rational functions to the sample ECDF with an optimization tool specialized for this case, Eureqa [SL13; SL09]. While not as good as a simple model with a probability distribution, having a rational function as a description for a dataset can still enable some further statistical and queuing theoretic evaluation.

2.4.7.2 *Statistical Tests*

To check the statistical goodness of the generated fits, statistical tests can be used. Generally, tests compare the values observed in an experiment 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.

First, as a simple measure, the *Pearson correlation coefficient* can be facilitated, comparing the covariance and standard deviation of the empirical and fitted variables. Another possible approach is *Pearson's χ^2 test for independence* [Pea00], which is the oldest known test and defined as

$$V = \sum_{i=1}^{k} \frac{(o_i - e_i)^2}{e_i}. \tag{2.7}$$

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 as the test for a given significance level. In most practical cases this comparison is conducted against precomputed tables with set significance levels. The 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 χ^2 test, albeit with a certain loss of precision.

Continuous data can be checked with the *Kolmogorov-Smirnov test*. First suggested by Kolmogorov in 1933 [Kol33] and expanded on by Smirnov in 1939 [Smi39] it is defined as

$$K_n^+ = \sqrt{n} \sup_{-\infty<x<+\infty} (F_n(x) - F(x)) \tag{2.8}$$

and

$$K_n^- = \sqrt{n} \sup_{-\infty<x<+\infty} (F(x) - F_n(x)), \tag{2.9}$$

for the ECDF $F_n(x)$ and CDF $F(x)$. 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.

Finally, diagrams of the empirical and fitted distribution — especially histograms, density, and CDF — should additionally be compared and checked for specific artifacts or outliers in a visual inspection.

2.4.7.3 *Random Sampling*

Most of the evaluations in Section 3.1 use random sampling to work on a subset of the original data. Not only does this simplify the handling of a dataset this large sets — working on a set with two billion entries can be quite problematic — but can even improve statistical significance, as rare outliers tend to get removed by drawing samples. By selecting entries using a uniform distribution it is ensured that no unintentional sampling bias occurs. Through a technique called bootstrapping, 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.

2.5 Core Network Architecture Summary

The chapter served as an introduction to mobile architectures, through a broad overview of the 3G mobile core network control plane, and the evaluation methodology, including a description of the dataset. Modeling mobile networks cannot be achieved without first understanding many of the aspects and protocols unique and intrinsic to mobile networks. And these differ a lot from the conventional wisdom found in wired network architectures. The determined definition of a control plane load serves as an essential distinction to regular mobile network investigations, which usually consider just the user plane. The gained knowledge will be utilized as a basis for the evaluations in the following chapter.

3

EVALUATING MOBILE SIGNALING TRAFFIC AND LOAD

With the architectural and methodological overview concluded this chapter can now move on to the actual evaluation. To this end, the previously described dataset is explored for any signs related to control plane load. The evaluation, including a statistical analysis, is given in Section 3.1. The measurement data backs up a number of assumptions on the behavior of different device and operating system types, but also reveals some remarkable signaling characteristics.

The results of these measurements are then used to construct a queuing theoretic load model for a CN in Section 3.2. This model is then extended with virtualization modifications to it and followed by a numerical simulation in Section 3.3 to confirm the viability of the models.

3.1 Mobile Core Signaling Evaluation

Finally, the core network control plane load evaluations can now be tackled. The previously described dataset is thoroughly investigated several approaches to measure load and related factors are iterated.

3.1.1 *Traffic Ratio Estimations*

To get a first grasp of the dynamics present in the dataset and the core network under investigation, Table 3.1 shows a small survey of the traffic composition split up by device type and OS categories. The majority of signaling messages originated from smartphones, which in turn generated only a small portion of user traffic when compared to 3G dongles.

With these numbers, the notion of active devices or tunnels can also be introduced. This only includes entities that, besides signaling, actively generated user traffic during their life cycle. Interestingly, only about 82 % of all unique devices in the trace were active and could be associated with at least one traffic flow. The remaining 18 % of devices still had an open GTP tunnel but never used it. This is a probably unexpected source of load for the core network, as it causes a significant amount of control plane load without any actual benefit to either the network or the device. The active device distinction will also be used later on in the evaluation.

Unfortunately, the dataset does not contain any hard numbers on the data volume of the signaling messages, which could be a direct indicator of the network load the control plane imposes. But using the estimated upper limit of a GTP message from Section 2.1.5, a rough upper limit on the total signaling traffic can also be derived. The following formula is used:

Table 3.1: Relative device-discriminated traffic statistics extracted from the dataset.

	Flows	Traffic	Tunnels	GTP messages	Devices
By device type					
Smartphones	20.58%	12.81%	60.31%	75.99%	37.97%
Regular phones	0.26%	0.37%	5.40%	0.94%	9.25%
3G dongles	66.55%	75.12%	12.71%	9.53%	25.10%
By OS					
Android	10.82%	6.48%	14.33%	43.33%	14.01%
iOS	7.22%	4.47%	18.91%	20.35%	7.94%
Symbian	1.02%	1.09%	21.17%	4.51%	12.97%
Blackberry OS	0.07%	0.10%	2.17%	2.60%	1.48%

$$v_s = 2S(v_{gtp} + v_{udp} + v_{ip}), \tag{3.1}$$

$$t_r = \frac{v_s}{v_t} \approx 0.7\%, \tag{3.2}$$

with the signaling volume v_s, the set of signaling messages S (times two since requests and responses are considered), the estimated size of a GTP message v_s, and the length of the UDP and IP headers. In this scenario, the traffic ratio t_r of v_s compared to the total traffic v_t is calculated to be a minute 0.7 %. Therefore, it can be that the volume of control plane traffic is not a limiting factor in the core network. Therefore, any impact on CN performance likely stems from other, load factors at the network nodes, 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.

Consequently, the following evaluations are all intended to find some indirect approach to measure the system's load.

3.1.1.1 *GTP Tunnel Duration*

The first indirect evaluation target will be the duration of the GTP tunnels. This duration is directly related to the amount of tunnel management signaling occurring between the SGSN and GGSN. In turn, each of these signaling interactions causes processing at the two involved nodes and changes the amount of state and its properties in the form of the PDP context. In terms of signaling messages, the tunnel duration implies both tunnel create and delete messages, but no update message.

For the purpose of the evaluation the duration is defined as the interval between corresponding GTP create and delete messages. As soon as the GGSN sends its successful response to the create request, it can be expected that the necessary state has been created throughout the CN and that the network is ready to forward user packets. Similarly, after a delete message, user traffic should not be forwarded anymore. However, state may still exist and could be freed up lazily. But the latter depends entirely on the specific implementation as it is not explicitly defined in the specifications.

As a side note, while the trace itself is only one week long, information on tunnels longer than this period can still be obtain when they were closed during the period. The trace's record on delete messages also contains the timestamp of the initial tunnel creation.

All the individual tunnel durations in the dataset are differentiated using two factors based on the presented TAC mechanics. The first part of the investigation looks at tunnels from different device types. After that, possible influences from the operating system are investigated.

Influence of the Device Type

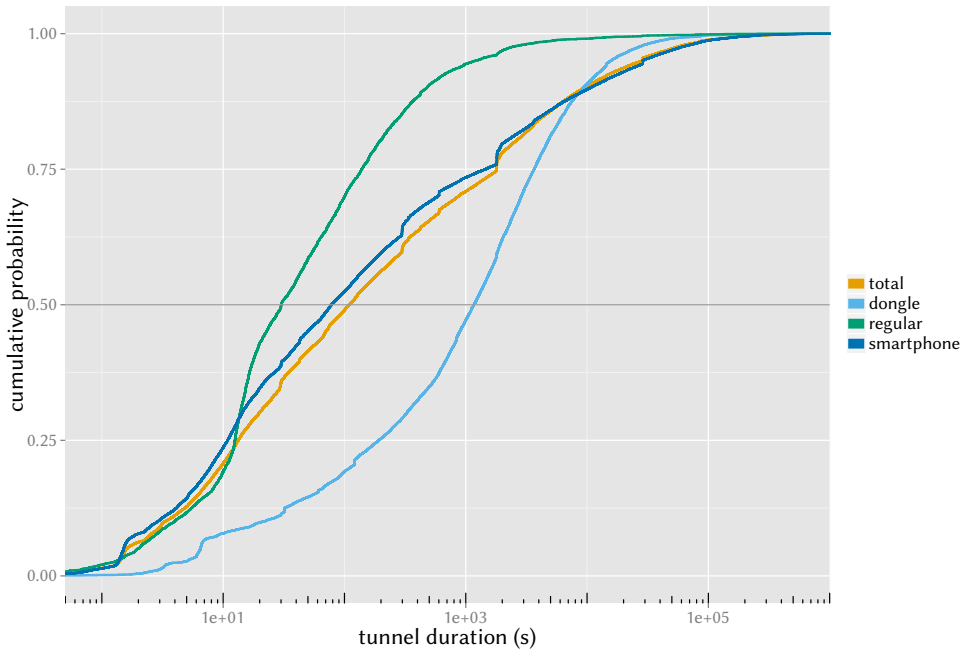


Figure 3.1: Tunnel duration distribution, separated for 3G dongles, smartphones and regular phones with medians at 115 s (total), 31 s (regular), 82 s (smartphone), and 1207 s (dongle).

Figure 3.1 shows the ECDF for the user tunnels and their PDP Context durations in the dataset. In this first graph, the duration of different device classes is distinguished and put in perspective to the overall duration distribution. The devices classes here are smartphones, regular phones,

and 3G dongles. It can be observed that tunnel durations range between mere seconds and more than one week.

The median can be clearly differentiated between device types, being much longer for 3G dongles than for mobile phones. This reflects expected user behavior very well and gives a first indicator on a possible influence of the user plane on the control plane.

Dongles are usually used with laptops to be able to work while being mobile. Therefore, dongle sessions last often for extended periods of time, longer than a few minutes and up to several hours. Also, this type of device is usually put into a standby mode after the period, which completely disables any mobile connections — and therefore any associated tunnel — instead of switching to low power radio idle modes. This is reflected in the dongle tunnel duration here as well. When compared to the other device category, dongles are more compactly centered around their median of about 20 min.

A similar behavior can be observed in the regular phone distribution with values arranged tightly around the median of 31 s. Compared to today's smartphones, data connections on regular phones are mostly explicitly initiated by user interaction, for example through starting a browser and viewing a web page, and generally do not generate traffic in the background. This could also explain the comparatively low durations observed here.

The picture is rather different in the smartphone tunnel duration. Here, often background tasks are running over long periods of time and devices try to keep connectivity up as long as possible (while still attempting to conserve power). Overall, this could lead to the smoother distribution seen here with no clear center value.

Overall, a relatively high number of tunnels with a duration shorter than 10 s can also be observed. Especially the peak at about 1.5 s — which is interestingly shifted to 6.8 s in the dongle distribution — is of note. This is even shorter than the default values for the RRC idle state transitions which causes the tunnel to be destroyed. It can be conjectured that these short tunnels have been explicitly removed by the UE as no other involved state machine has timers this short.

Another distinct step in the total and smartphone distributions can be observed at the 30 min mark. As it is only present in these to categories — and the total distribution looks to be mostly governed by smartphones — it is reasonable to assume that the cause for this is a specific behavior observable in some aspect of smartphone related influence factors.

Influence of the OS

Next, the smartphone and regular phone categories are further broken down by their OS. Only the three major systems, Android, iOS, and Symbian, are identified here, the amount of other types was negligible. The smartphone category is almost exclusively represented by Android and iOS devices, while Symbian devices make up most of the regular phones but is also represented in a number of smartphone models.

Figure 3.2 depicts the ECDF of the tunnel durations of these categories in relation to the total duration distribution. They immediately exhibit a clear difference between individual OSs.

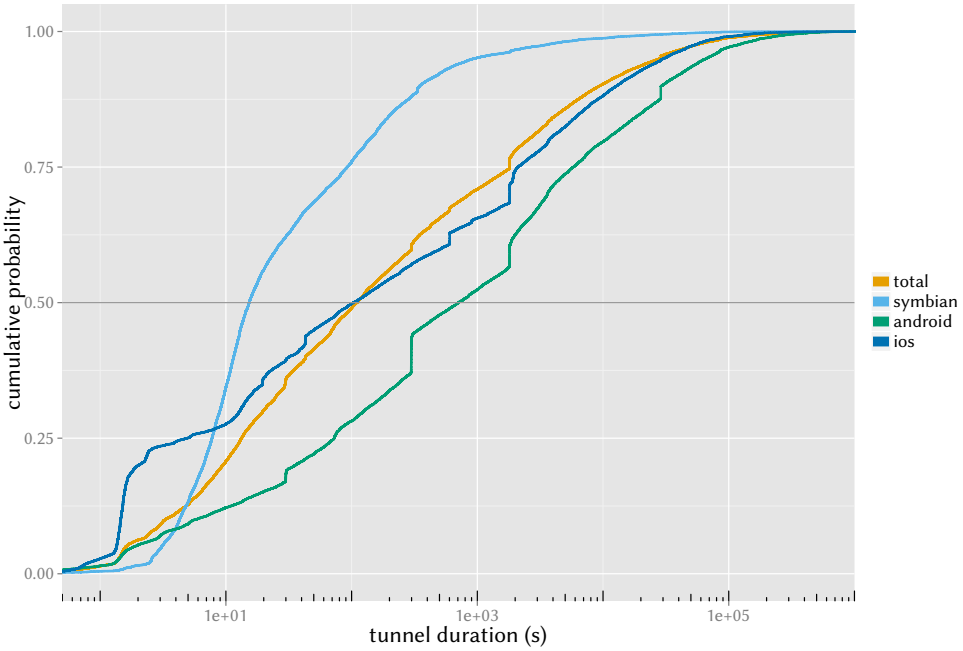


Figure 3.2: Tunnel duration CDF, separated for select OSs; Medians at 115 s (total), 15.5 s (symbian), 104 s (iOS), and 765 s (android).

The Symbian tunnel durations are similarly distributed to the previously depicted regular phone category, albeit with an even shorter duration median of about 15 s. This is an indicator of the large intersection between these two groups and the explicit user traffic property attributed to regular phones.

The two smartphone-exclusive OSs have remarkably similar tunnel distributions with the exception of the Android tunnel distribution shifted to much longer tunnels. This is mostly due to the larger accumulation of iOS tunnels around the previously mentioned 1.5 s mark. Over 20 % of all tunnels established by iOS devices are shorter than 2 s. A possible explanation is an interaction between the described implicit background traffic happening in intervals and the efforts of iOS phones to preserve as much energy as possible.

To this end, phones aggressively force their radio connection to the low power idle states or even completely shut off the radio immediately after transmission have ended, circumventing RRC timers. To achieve this, iOS devices are known to implement a form of 3GPP Fast Dormancy [GSM12]. It is deemed to improve device battery life, radio signaling and radio spectrum efficiency. Due the more frequent state transitions it also could cause an increase in core network tunnel management signaling, which is probably what happened in the iOS case depicted in the ECDF.

Another set of tunnel duration accumulations are also visible in the OS distributions. Two types of steps should be distinguished here. First are accumulations that occur across multiple or all categories. This points to an influence source outside of the specific category. If the artifact

is present in every distribution it is even likely that the source is a behavior of the network's state machines. The second type of accumulation is local to one or some categories, which places the root cause into the region of these categories and their related influence factors.

In case of the OS category, additionally, peaks at 30 s, 300 s, and 600 s can be observed. However, whether this behavior can be attributed directly to the operating systems themselves cannot be decided just by looking at these distribution. Other factors, e.g., the device's baseband and user traffic dynamics, also play a role.

Influence of the Time of Day

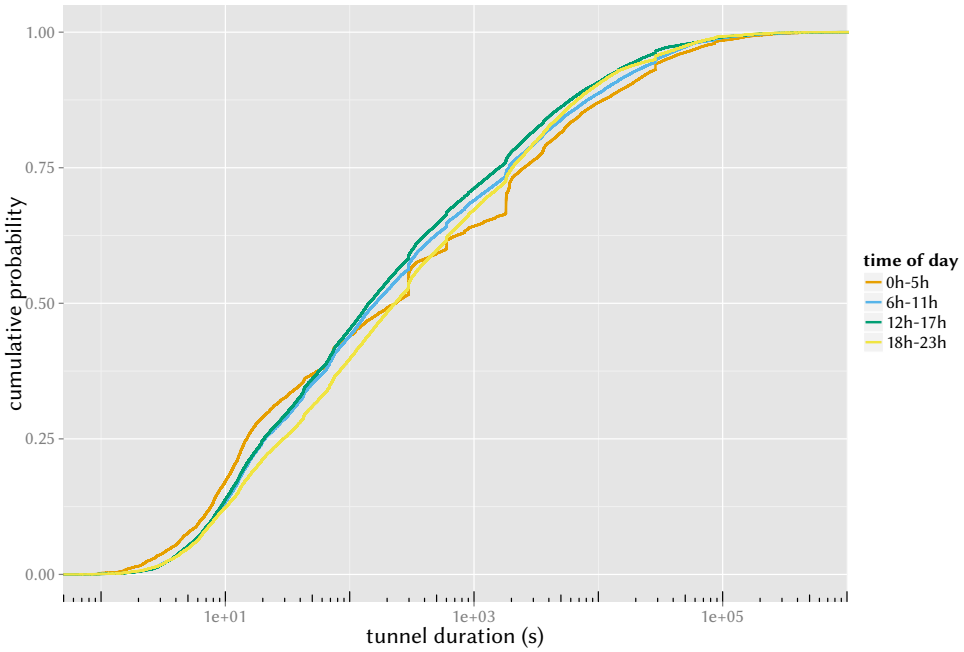


Figure 3.3: Tunnel duration of all active tunnels by time of day.

In addition to device factors, diurnal effects could also play a role in the duration of tunnels. Figure 3.3 depicts individual ECDFs of the tunnel duration for four six-hour intervals of a day, starting at midnight. While no clear distinctions are visible, there is a trendto shorter tunnels in the early morning hours. The early afternoon hours tend to produce tunnels more centered around the middle duration range. Even longer tunnels should be treated with reservation, as they exceed the length of their assigned time slot in the ECDF and span a larger time frame. Only the tunnel creation point is guaranteed to be in the slot.

Influence of Other Factors

Due to the nature of the trace dataset at hand many other influence factors are hard or outright impossible to distinguish. Some factors are unknown from the CN perspective, as the mentioned device baseband, while others have not been recorded in the trace.

For example, it would theoretically be possible to investigate the influence of the RAT as it is an IE in the GTP messages recorded in the trace. The radio access parts of GSM and UMTS are completely different — including the RRC state machines which were depicted in Figure 2.5 — and therefore could also differ in their control plane load impact on the core. However, the RAT IE is optional and only set in less than 1 % of the available records. As the radio access can change even during an existing tunnel — in which case the GGSN receives a GTP update request informing the node about the change — a complete picture without gaps in the knowledge of the RAT would be required to do any investigation on this.

Influence Strength of the Categories

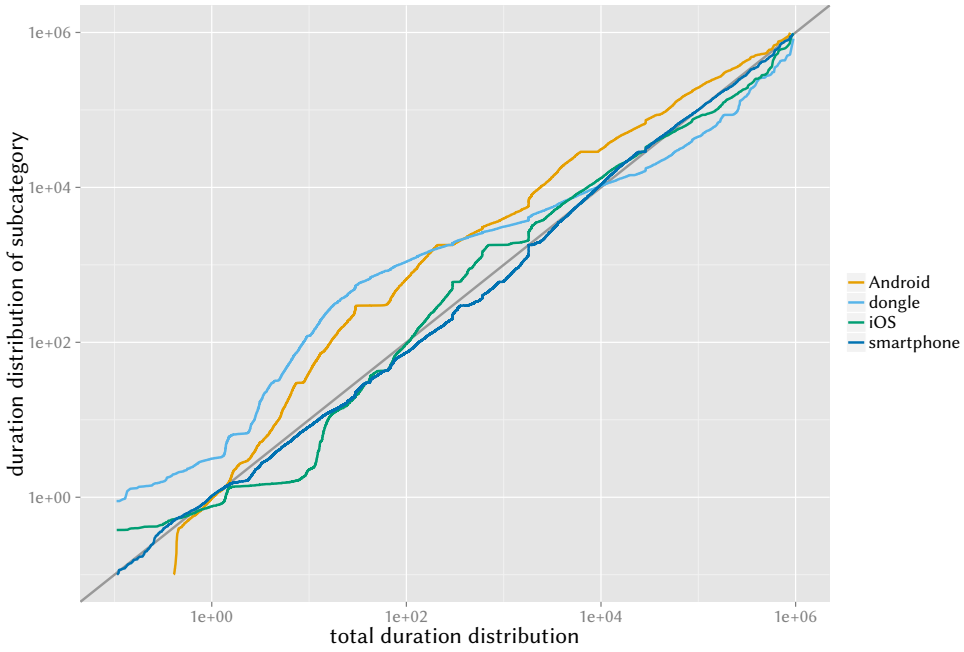


Figure 3.4: Q-Q Plots of the tunnel duration distributions in comparison to device classification categories.

To ascertain which of the investigated device categories influences the total duration distribution most, Q-Q plots are created and investigated. It is conjectured that the amount of influence on the duration distribution is correlated to the influence on the control plane load. In theory, if both durations follow the same distribution, one expects a straight diagonal $y = x$ line through

the origin. A steeper incline indicates more compact regions in the distribution plotted on the x axis and vice versa.

The Q-Q plots in Figure 3.4 compare the total tunnel duration distribution to the duration distribution of the dongle, smartphone, Android, and iOS classification cateogries. It can be observed that the smartphone duration distribution is distributed almost equally to the total except for minor variations. However, the 3G dongle tunnel durations follow a very different distribution. Their effect on the total duration distribution seems to be negligible despite the large amount of traffic they are causing. This is also a first indicator that smartphones might have a larger impact on signaling than other device types.

Looking closer at the smartphone category, Q-Q plots of the two major OSs are investigated. With the exception of the large sub-2 s peak in the lower tail of the distribution, iOS device tunnel durations are very similar to the overall tunnel duration distribution. The same can not be said about the Android distribution, which deviates somewhat in the distribution's center but is similar to the total distribution in the upper tail. Even devices with just a different OSs seem to strongly differ in their influence on duration distribution and therefore on signaling.

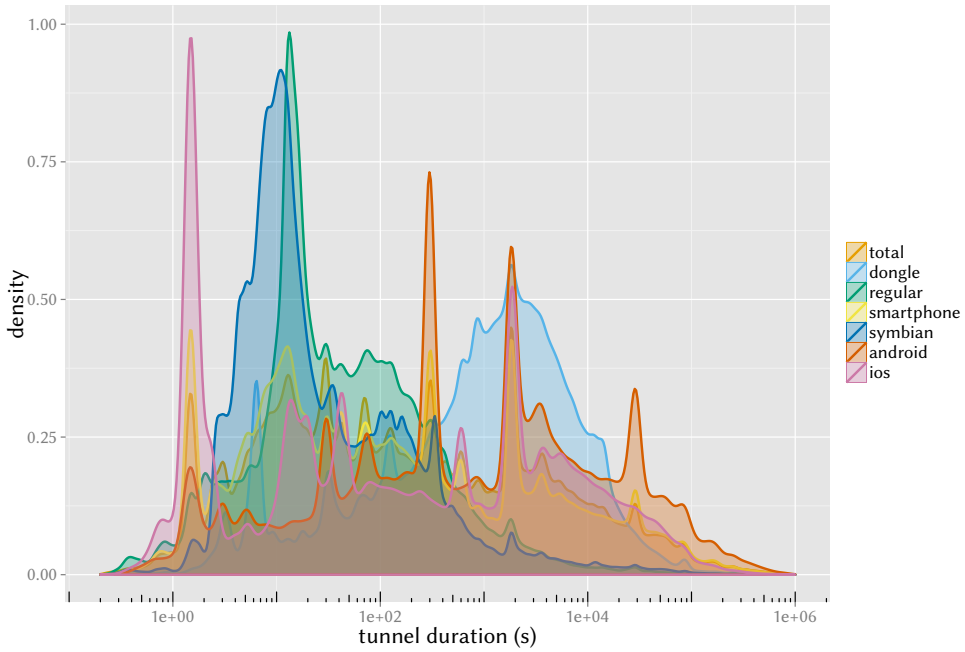


Figure 3.5: Logscale density plot of the tunnel duration with all classifications.

Figure 3.5 attempts to depict where in their distributions the investigated device categories show the most impact on the total distribution. The plot shows the density of all previously investigated device influence categories.

It is evident that the durations are not evenly distributed, but rather follow sharp spikes. One of largest spike across all categories is the one at a duration of 30 min, with about 1.8 % of all tunnels in the network falling into that region. Since this spike happens across all device types,

it makes a rather strong case for being induced by the network. On the other hand, the bulk of tunnel durations in the short-to-medium range does not seem to be governed by the two major smartphone operation systems but by other devices in the network, which do not show major spikes in other bins.

Besides the long-tailed behavior in the upper tail of the tunnel durations another slight accumulation effect, repeating itself every 6 d to 7 d, is present in the upper tail. This phenomenon is as yet of unknown origin and does not coincide with any known timers of the 3G mobile network.

The investigation of this data leads to the conclusion that the planning and dimensioning of the control plane needs to watch the behavior of smartphones more carefully than that device types.

3.1.1.2 *GTP Tunnel Arrivals*

The duration of GTP tunnels is but one aspect of influence on control plane load. The arrival process of these tunnels is also interesting in itself, specifically, the arrival of tunnel requests, i.e., GTP create requests, at the GGSN.

An arrival process can be described in two distinct ways. First by the number of arrivals in a given time interval. Second, by the Inter Arrival Time (IAT), the time between two consecutive arrivals. Depending on the choice one has to deal with either a discrete or a continuous distribution.

Here, the tunnel arrival process is investigated with both approaches. This also adds to the foundation of the load model constructed in the next section. Note, that the notion of classifying arrivals into influence categories based on device specifics is omitted here. An investigation of this process can not be realistically be conducted categorized and still relate to the total system load.

Figure 3.6 depicts a histogram of the number of tunnel arrivals per second during the whole trace duration period. Of note is the clear bimodal nature with one peak around twelve arrivals per second and the other in the low thirties. While the distribution is rather compact around these two peaks, there are some outliers reaching 107 arrivals per second. If the hypothesis of the correlation between signaling load and number of arrivals holds, it can be assumed that load is not constant but rather switches between two modes with some periods of very high load induced by an increased number of arrivals.

A reasonable cause for the occurrence of these two modes can be found in the diurnal arrival patterns. Figure 3.7 contains a violin plot of the tunnel arrivals. This type of plot is similar to a box plot but additionally shows the density of the individual items on the vertical axis. Here the arrivals are broken down to hourly slots.

The nocturnal plateau of arrivals between midnight and 05:00:00 and the longer daytime plateau between 08:00:00 and 19:00:00 match the two modes found in the histogram. In between are short transition phases. The density of the arrivals during daytime indicates a spread of the number of arrivals over a larger range. This could be an indication of load fluctuations in the system.

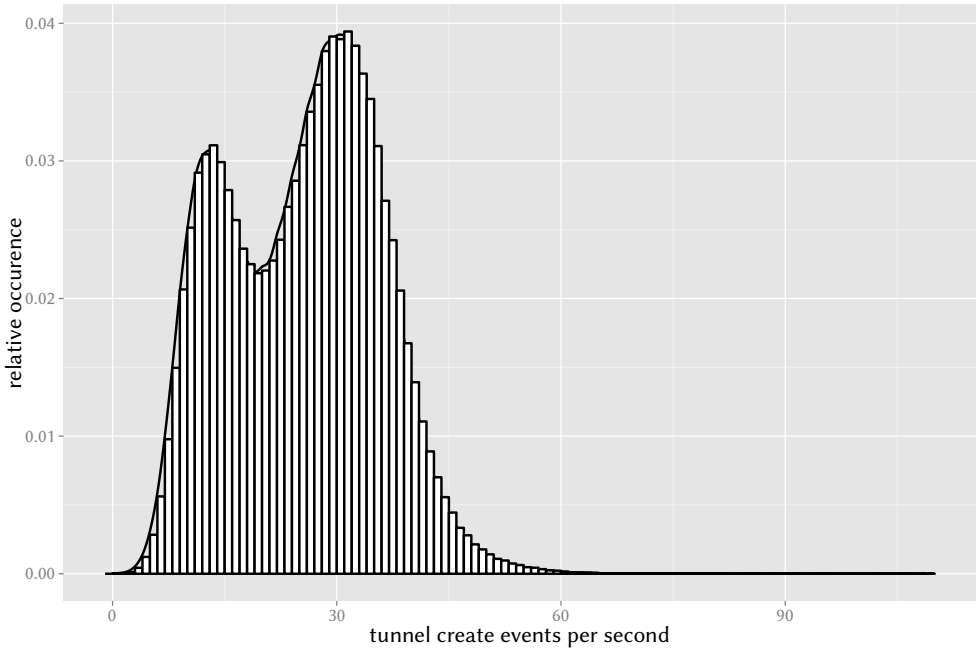


Figure 3.6: Tunnel arrivals histogram overlaid with a density plot.

Complementing the arrival rate evaluation is the investigation of the tunnel IAT. This metric is more sensitive to short time fluctuations of arrivals and more suited to describe the arrival process for use in the proposed load model.

The overall picture of all arrivals is given in the ECDF of Figure 3.8a, again broken down by time of day. Obviously the same previously observed diurnal load oscillation can again be perceived. The median IATs fall in the range of 20 ms and 60 ms, enveloped by the distribution for the hours starting at 16:00:00 and 02:00:00 having the lowest and highest IAT respectively. Additionally, tunnel arrivals are occurring at an increased frequency with an interval of multiples of 20 ms, which generates these wave-like steps in the ECDF plot. As this is happening very regularly at every time of the day, a source inside the mobile network is indicated.

A hypothesis as to the origin of this effect is the value of the Transmission Time Interval (TTI). This property determines the duration of a mobile network's radio transmission slot. In 3GPP standards up to UMTS the default value of the TTI is either 10 ms or 20 ms, newer versions of the specification additionally allow values of 2 ms (in HSPA) or even 1 ms (for LTE). The absolute time of every transmission slot is also synchronized across every base station in the whole mobile network, which makes the TTI noticeable even when not measuring directly at the radio link.

The observed step-width of 20 ms therefore indicates that the tunnel establishment signaling procedure 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 gives the arrival process batch properties. As a

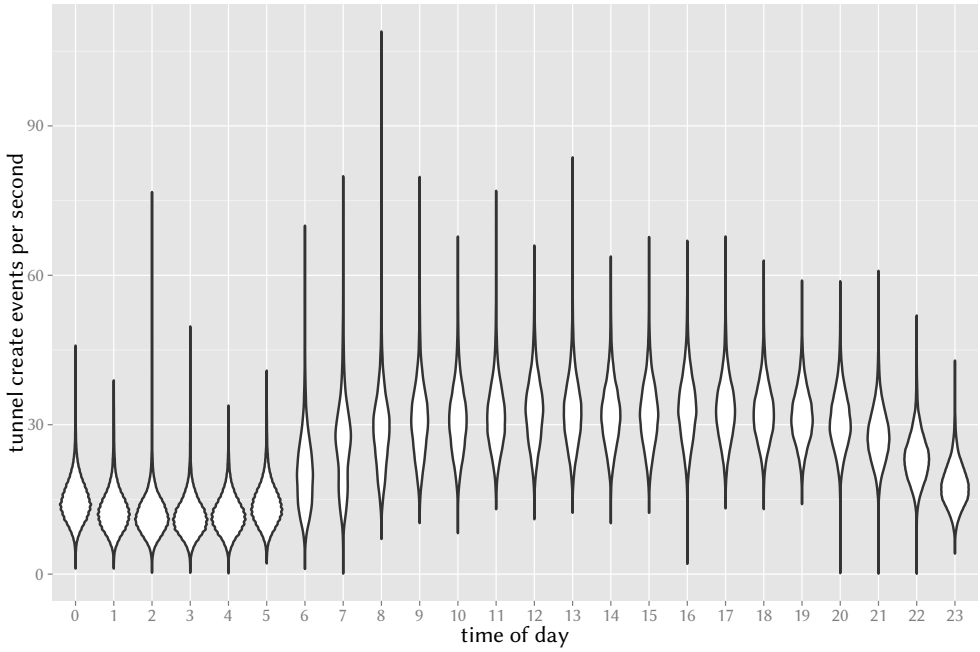


Figure 3.7: Violin plot of tunnel arrivals in one second per time of day.

result the load at the GGSN increases momentarily when a batch arrives. The GGSN would then need to process more requests simultaneously than if the arrivals followed a smooth stochastic distribution.

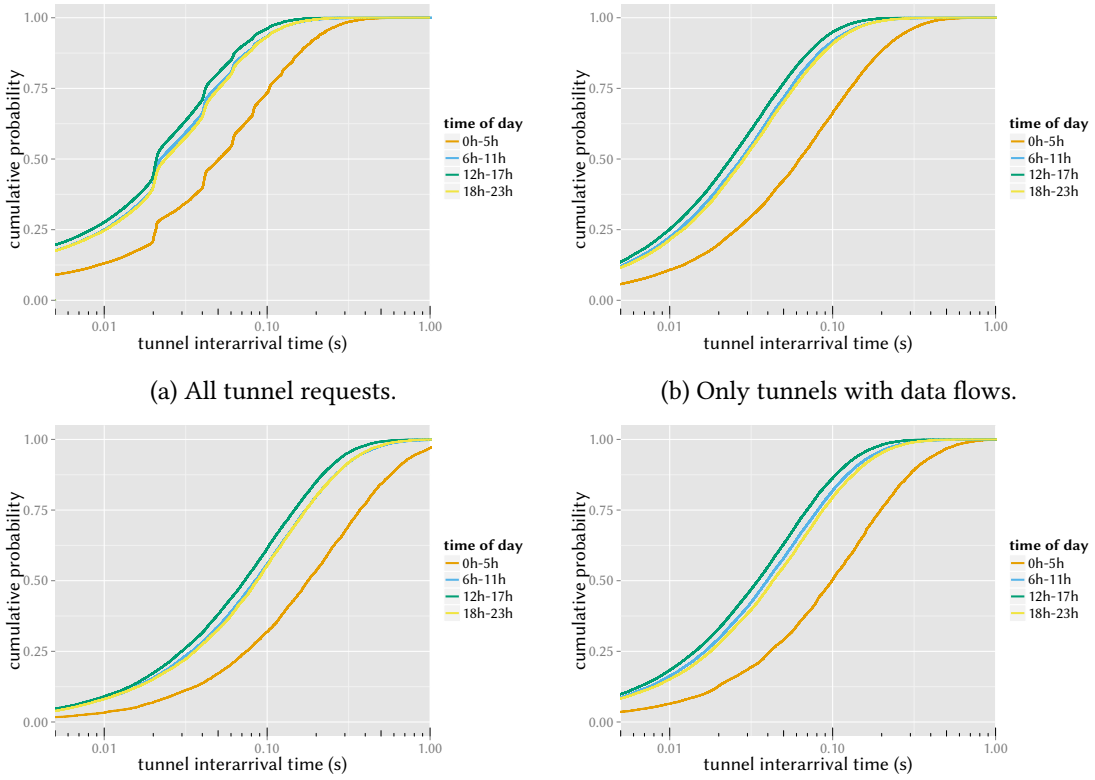
This effect becomes more peculiar when the tunnel arrivals are further broken down. Figure 3.8b only displays arrivals of tunnels that actually transported user traffic during their lifetime. Here, the influence of the effect is visually unnoticeable. This could be attributed to the fact that most active data connections during the time of the trace recording were already using almost exclusively HSPA or better, which sees the much lower TTI. Only older, regular phones establish plain UMTS connections and often do not even use it.

The discrimination of the IAT distribution by RAT that was used during the creation of the tunnel reveals no further information. Due to the much lower number of connections the GPRS distributions are shifted to much higher intervals than the UMTS specific distributions.

3.1.2 *GTP Tunnel Message Processing Time*

Finally, the GGSN's processing time of GTP tunnel management messages is investigated. Potentially, this can be a direct measure of the load at the node. In times of higher load one would expect a higher processing time of signaling messages.

From the network trace the processing time can be calculated by two timestamps in each record. As the trace is recorded at the Gn interface these timestamps represent the points in



(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 3.8: ECDFs of the tunnel IAT in seconds by time of day.

time the GTP signaling request and subsequent response pass on the link to and from the GGSN. Therefore, they can also be interpreted as the start and finish of the involved processing at the GGSN.

Generally, the processing time of all three message types — i.e., creates, deletes and updates — could be calculated. It would be of special interest to know what influences the setup time of tunnels, 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, due to unrecoverable issues with the recording of the dataset, the related timestamps for both the create and delete messages records were completely unreliable and did not allow for an investigation of the processing time.

Only GTP update messages were unaffected by this mishap and gave the opportunity for further investigation. The trace contains roughly two orders of magnitude more update messages than either creates or deletes, spread out almost evenly over the whole observation period. Therefore, a node load investigation should still be possible with just the updates messages.

Figure 3.9 depicts a band of ECDFs for the processing time of update messages by time of day. The processing time distribution almost perfectly follows a continuous uniform distribution

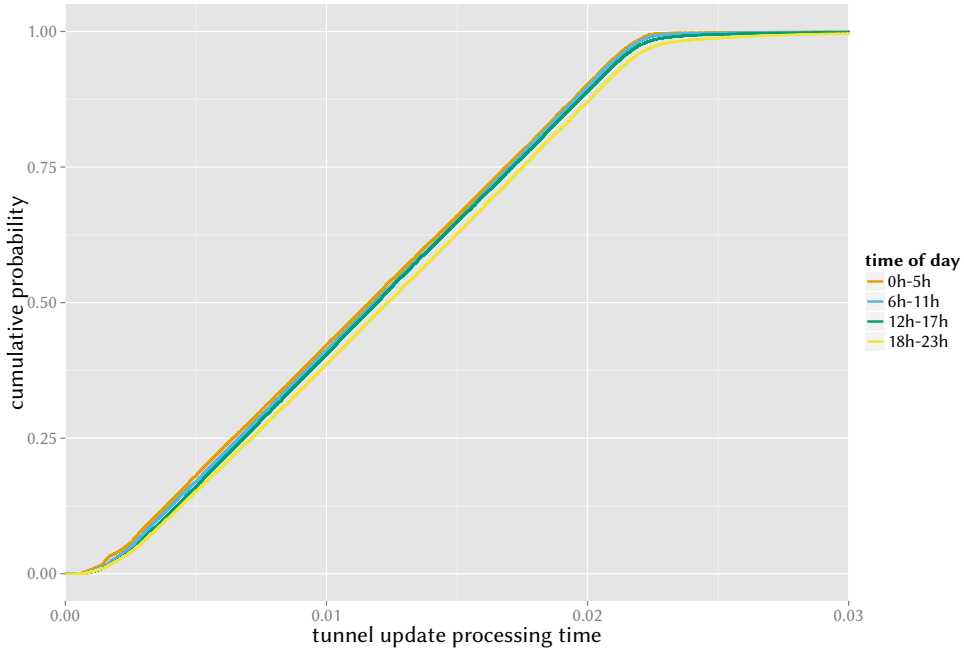


Figure 3.9: ECDFs of the time in seconds it takes a GGSN to process a GTP update event, separately plotted for four time slots each day.

between 2 ms and 22 ms. Only the upper end displays a slight long-tail behavior. The impact of the time of day is very slim with slightly higher processing times during the evening, the same time frame which also experienced an elevated arrival rate.

The occurrence of a continuous uniform distribution is rather unexpected as these 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., process scheduling or other queuing artifacts. The source of this effect is still unknown and the current dataset does not allow for a more thorough investigation. Still, the fact that a higher update processing time coincides with an increase in the arrival rate points to an influence of tunnel messaging on the load of a GGSN.

3.1.3 *Statistical Evaluation and Data Fitting*

The uncovered empirical distributions for both the tunnel duration and the tunnel IAT are now to be matched against theoretical probability distributions. Therefore, a univariate distribution fit to the experimental data was conducted. Having a concise representation for the empirical data will help in creating a model of the core network, which is the task conducted in the sections following after this.

IAT Fitting

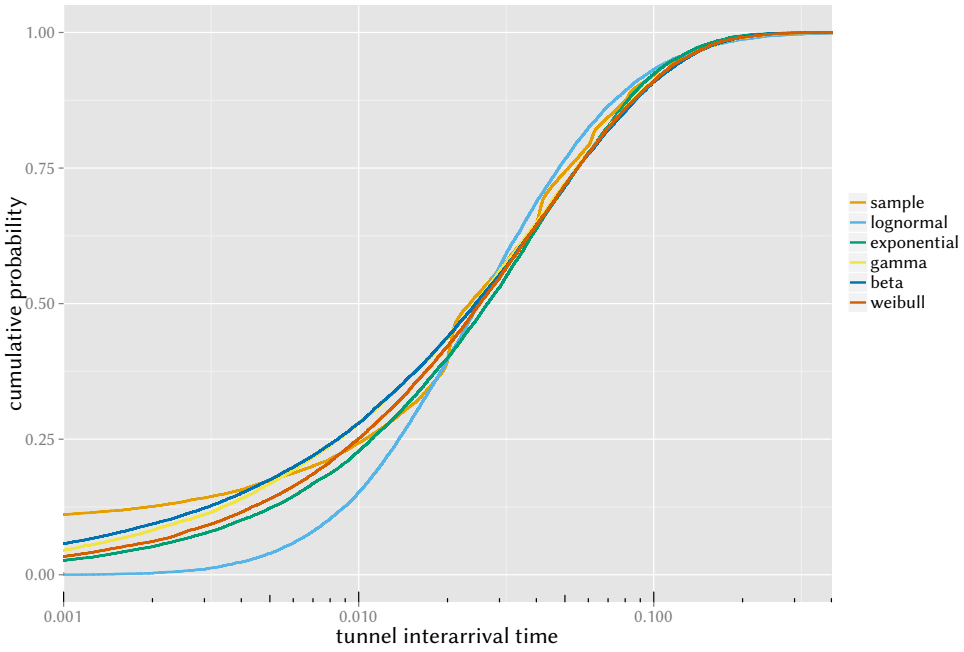


Figure 3.10: Sampled inter-arrival time CDF and fitted theoretical distributions.

In order to investigate the tunnel IAT, Figure 3.10 displays the overall ECDF with fits for various basic probability distributions. Each of the fits was generated through the method of moments matching.

The goodness of these fits was checked both visually using the CDFs plots and numerically with goodness of fit measures, using Pearson's correlation coefficient and Pearson's χ^2 test. Unfortunately, none of the probability distributions reaches the significance level for χ^2. This can probably be largely attributed to the various previously described artifacts in the data. Matching them visually, the exponential fit seems to be reasonably close to the experimental data.

To improve the fits, two modifications were made to the process. First, to remove the 20 ms steps, only the active tunnels were taken into consideration. Second, the overall IAT distribution was once again split up into time of day slots. The overall distribution is just a superimposition of the individual slots anyway. Therefore, this should further improve the fidelity of the fits.

The results are depicted in Figure 3.11. To improve plot visibility only four larger time slots are displayed here while the actual fits were conducted for one-hour slot. Parameters for the exponential distribution $F(x) = 1 - e^{-\lambda x}, x \geq 0$ and the corresponding correlation coefficients to the original data for the four time slots are given in Table 3.2. The fitted functions match the empirical data quite well, with some deviation present at the left tail but an overall positive correlation coefficient approaching 1.

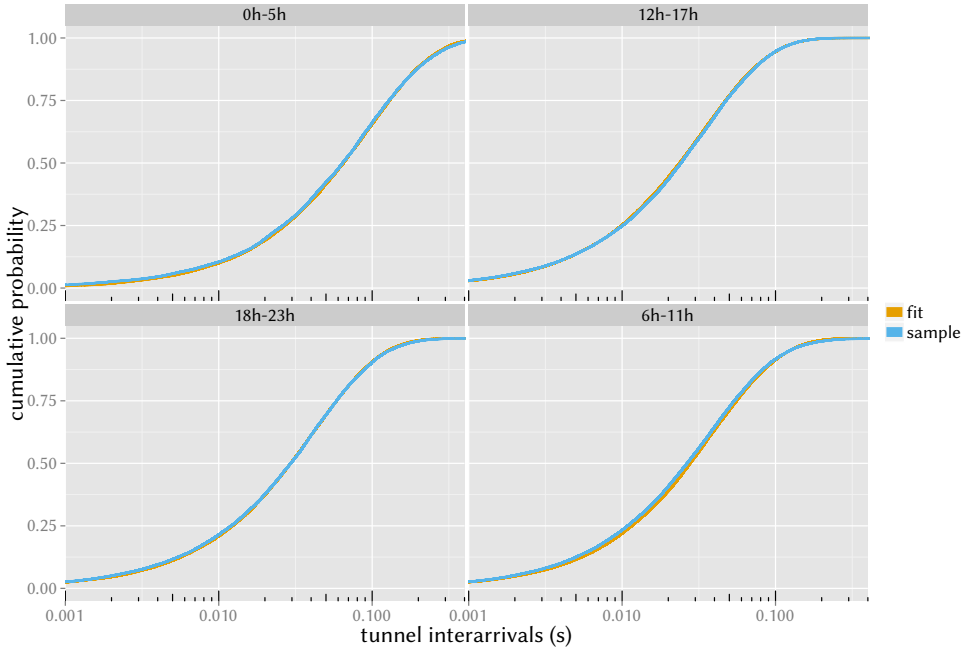


Figure 3.11: Empirical and exponentially fitted CDFs of the tunnel IAT by time of day. CDFs are overlapping as the coefficient of determination is close to 1.

Duration Fitting

The second fitting effort surrounds the empirical data concerning the tunnel durations. However, none of the basic probability distributions (including exponential, gamma, and Weibull distributions) fit the tunnel duration even remotely. One of the reasons for this is probably that the tunnel duration is influenced by an overwhelming amount of factors, which were previously described. This superposition, especially with the user behavior, will result in unpredictable results that does not follow any basic probability distribution.

Table 3.2: Parameters for the exponentially distributed inter-arrival times and corresponding Pearson correlation coefficients.

Time of Day	λ	$R_{arrival}$
0h-5h	10.67477	0.995
6h-11h	24.53298	0.992
12h-17h	29.2504	0.993
18h-23h	23.49983	0.986

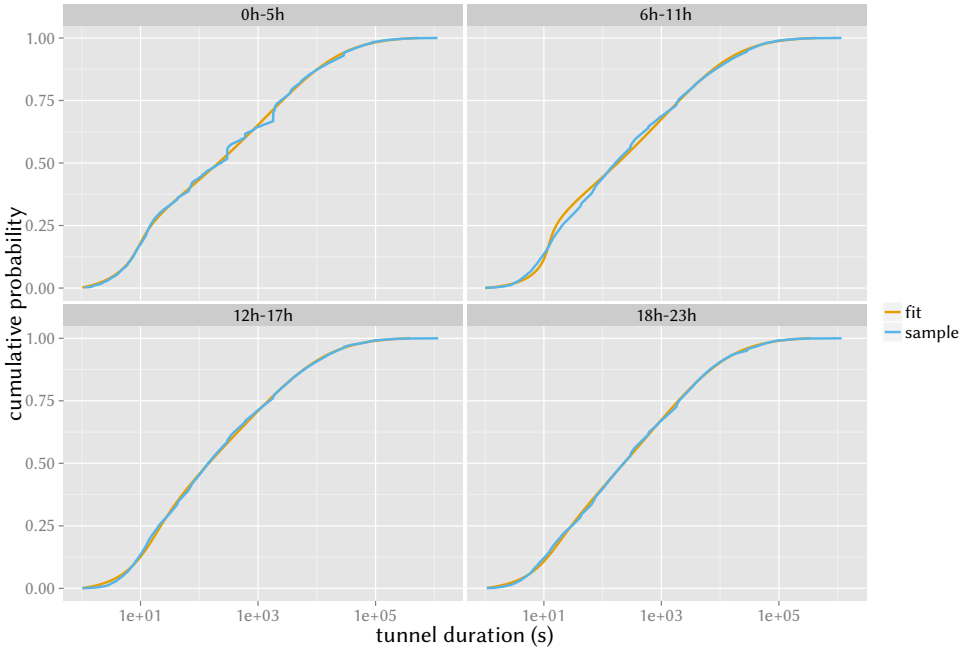


Figure 3.12: Empirical and fitted CDFs of the tunnel duration by time of day with fitted rational functions.

Instead, rational functions are fitted to the ECDFs using the proprietary third-party tool *Eureqa* [SL09; SL13]. This allows for a much closer fit as seen in Figure 3.12, but limits its application in the statistical evaluation.

Table 3.3: Inverse rational functions fitted to the ECDFs of the tunnel duration by time of day and correlation coefficients of the fit.

Time of Day	**Inverse Serving Time CDF Representation**	R_{dur}
0h-5h	$0.919 - 60.614y - 3498.78y^3 - \frac{110.707y+2289.94y^3}{y-1.005}$	0.999
6h-11h	$1 + 117.484y - 368.643y^2 - \frac{1720.13y^4}{y-1.004}$	0.999
12h-17h	$0.953 + 69.491y + \frac{81146.1y^3+1.086\times10^6y^5}{805-802.01y}$	0.999
18h-23h	$0.912 + 82.056y - \frac{2936.93y^4}{1.945y-1.953}$	0.999

Table 3.3 contains the functions which were fitted to the *inverse* CDF. The inverse was chosen here to simplify the modeling and simulation process coming afterwards. The functions can be easily inverted again for other purposes. Both the CDFs in the plot as well as the Pearson correlation coefficient, which again approach 1, confirm the goodness of the fitted functions.

3.2 Modeling Mobile Network Load

The next logical step after the collection of empirical data and the execution of a statistical analysis lies in the creation of models abstracting this real system. While some loss of precision is incurred, models are much more flexible and can have numerous applications. Load models and the derived information on the network QoS parameters can serve as a basis for the video measurement framework of Section 4.4, which uses arbitrarily chosen values for its latency and loss experiments. By utilizing the load model a more realistic mobile network could be emulated. Additionally, network operators can also be supported in predicting the signaling load in their core network with the benefit of improved network engineering and correctly scaling core components.

On the basis of the tunnel distributions attained in Section 3.1, models for both a traditional GGSN as well as a virtualized GGSN are introduced. The performance trade-offs when using a virtual GGSN are further studied, discussing different options to consider when using the virtual node.

The modeling and simulation of the resulting models was conducted in cooperation with the University of Würzburg and partially published in [FSH14].

3.2.1 *Queuing Theory Basics*

To understand the modeling process some knowledge on queuing theory is required. The next few sections give a short overview on the definitions used in the subsequent sections.

3.2.1.1 *Little's Law*

A basic queuing system can be expressed as a stream of customers arriving at an arbitrary system with a rate λ. This system then processes the customers, taking an average time of W on a number of processors until the customers depart again. On average L customers will be in the system. The representation — and queuing theory in general for that matter — was originally devised for telephone networks by Erlang [Erl17]. From that, Little's Law [Lit61] can be formulated as

$$L = \lambda W, \tag{3.3}$$

which holds universally, independent of any specific arrival or service time process.

3.2.1.2 *Kendall's Notation*

To distinguish the variations of a queuing system's parameter a simple convention and naming scheme was devised by Kendall in 1953 [Ken53] and later extended on. In its simplest form the notation reads $A/S/s$ with A denoting the arrival distribution, S the service time distribution, and s the number of servers. Here, an extended notation will be used,

$$A/S/s/q \tag{3.4}$$

which additionally describes the queue length q. With this naming scheme, a queuing system ($q = \infty$) can be easily distinguished from a blocking or loss system ($q = 0$). The most commonly used arrival processes and service time distributions are summarized in Table 3.4.

Table 3.4: 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

3.2.1.3 *Information Gain*

Depending on the complexity of the specific queuing system model, much information can be gained from an analysis of the given model. In simple cases the state probability can be mathematically determined, i.e., the probability that exactly m customers are in the system concurrently. If this number is higher than the number of processors, this also determines the queue length or the blocking probability p_B if there is no queue. Other properties include for example the waiting time of customers.

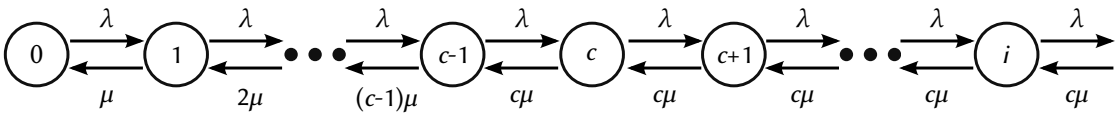


Figure 3.13: $M/M/c/\infty$ Markov chain model.

One such basic queuing system is $M/M/1/\infty$ [Kle75, pp. 94-99], on which stationary analysis can be applied upon. Both the one processor queue and $M/M/c/\infty$ can also be easily expressed as a Markov chain due to their memoryless property. Figure 3.13 depicts the state transitions of a system with c processors and a queue length of $i - c$.

More complex models are often not tractable by stationary analysis or other mathematical tools any more and no general solution is known. This is especially true for the class of $G/G/c$ systems, which can only be directly solved under certain conditions. System parameters may still be investigated using numerical queuing simulations. Hereby, both the arrival and the serving process are implemented in a DES using random numbers drawn from the desired distributions in order to ascertain the system load and blocking probability.

3.2.2 *GGSN Model Rationale and General Queuing Theoretic Representation*

The GGSN was already determined to be critical to the CN's load. Therefore, the network will be represented by this node in the model. Additionally, most of the load influencing factors are at least to a degree related to the GTP tunnels. So, to dimension a mobile network to its control plane load, the number of supported tunnels has to be modeled.

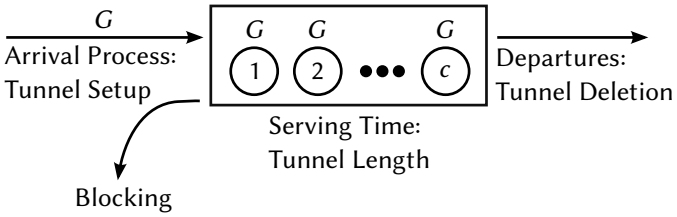


Figure 3.14: Queuing system representation of a mobile network's GGSN.

Figure 3.14 shows this model for the proposed tunnel load metric. It is in its generic form a $G/G/c/0$ system. Tunnels enter the system governed by a general random distribution and are served at the GGSN for the duration of their existence. This duration also follows a general distribution. Afterwards, tunnels leave the system again through the reception of a GTP tunnel delete message. If all c serving units are filled, blocking occurs and arriving tunnel requests are rejected.

The number of serving units corresponds to available resources at the GGSN. The maximum supported number of concurrent tunnels is hard to estimate as it depends on a number of factors, most of which are unknown for this modeling process. This could include soft-limits like the specific configuration, and hard-limits, e.g., the GGSN's processing and memory constraints.

For the purpose of creating an initial toy model the generic $G/G/c/0$ is simplified to a $M/M/\infty$ system. As stated, no actual limit to the number of virtual servers is known and the data also does not indicate any obvious limits. Thus, an unlimited system with neither blocking nor queuing is assumed for this simple model.

Now, assuming both a Poisson arrival and an exponential serving process (temporarily neglecting the fact that no basic function matched the GGSN's serving process), a stationary analysis can be conducted. As seen in the statistical evaluation, the former condition may hold, but the serving time is definitely not exponentially distributed. However, for the toy model this assumption is still made to get an initial grasp of the model.

The diurnal influences seen in the tunnel arrivals in the trace data are also temporarily ignored and only the overall empirical distribution is taken into account. Through distribution fitting with moment matching the overall arrival rate is set to be $\lambda \approx 25.641$ in the trace. The exponential service time distribution is calculated to have the parameter $\mu \approx 1.587 \times 10^{-4}$. Using Little's Law this gives an estimate for the mean number of concurrent tunnels at the GGSN in a $M/M/\infty$ system of

$$L = \frac{\lambda}{\mu} \approx 161.6. \tag{3.5}$$

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

3.2.3 *Representative GGSN Models*

With the experience from the toy model at hand more appropriate models can now be constructed to better accommodate for the core network's properties. Two models are provided here. The first describes a monolithic version of a GGSN, closely resembling the system used traditionally in the network. The second model is that of a hypothetical virtualized GGSN using Network Function Virtualization (NFV). In NFV [NFV13] custom monolithic network nodes are replaced by commodity hardware. The tasks solved by the original hardware are migrated to a pure software implementation.

3.2.3.1 *Monolithic GGSN*

The GGSN is considered to be one fixed monolithic entity, even if in reality it often consists of multiple servers. The entire GGSN is purchased from a vendor as a single entity with very little control over its inner workings for the operator. For example, idle instances can typically not be deactivated or reused for other purposes.

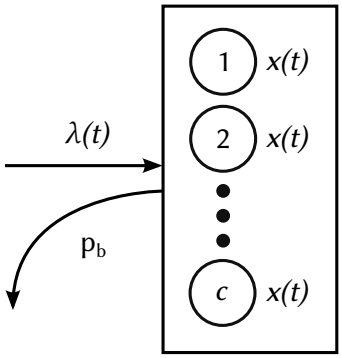


Figure 3.15: Traditional control plane load modeling approach to a GGSN.

The queuing theoretic equivalent is displayed in Figure 3.15 and is very similar to the basic toy model. New tunnels requests arrive according to a Poisson distribution with a rate of $\lambda(t)$ at the GGSN. The periodic time-of-day dependence of these exponentially distributed IAT and the corresponding distribution fits were extrapolated from the trace data.

Furthermore, the model has a maximum tunnel capacity of c. When this capacity is reached, blocking will occur and further incoming tunnels are rejected. The governing factors of the

capacity are mostly the node's available memory and processing capabilities. Monolithic GGSNs need to be preemptively dimensioned in such a way that blocking rarely happens, often resulting in gross overdimensioning as the node can not be easily scaled after it has been deployed.

When an incoming tunnel request is accepted one of the GGSN's serving units will be occupied for the tunnel's duration $x(t)$. Following the trace data, this duration is assumed to be of an arbitrary, non-Markovian service time distribution, again with a slight time-of-day dependence.

Combining the model with the exponential and rational function fits functions previously depicted in Tables 3.2 and 3.3 this results in a **non-stationary Erlang loss model**, or more precisely

$$M(t)/G(t)/c/0. \tag{3.6}$$

With this model, high control plane load can be indirectly described as the system's blocking probability p_B. The peak load can be ascertained by looking at the busy hour period where the arrival rate is the highest. No exact mathematical solution is known for this type of model. Only if the service time distribution can be confined to certain specific distributions, e.g. Phase-type distributions, some approximations can be made [DMW95]. However, the evaluated trace data does not give any indication of the presence of such a Phase-type in the service time distribution. Therefore, the model falls back to a non-stationary general distribution and a simulative approach to evaluate the model will be taken in Section 3.3.

3.2.3.2 *Virtualized GGSN*

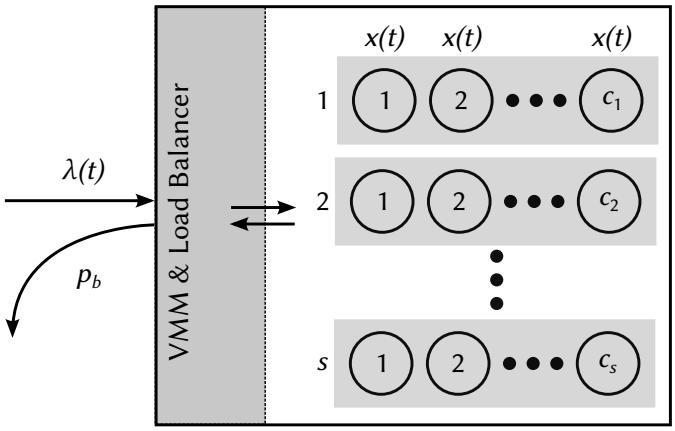


Figure 3.16: Model of a GGSN using NFV.

In the second model virtualization concepts are introduced. The assumptions of the non-stationary Markov arrival process $\lambda(t)$ and the serving time distributions $x(t)$ are carried over. However, instead of one server processing every tunnel, the system is now partitioned into individual server instances coordinated by a load balancer in Figure 3.16.

One virtual GGSN has up to s servers instances s_i. Each of the individual instances can be much smaller than the monolithic GGSN, having a concurrent tunnel serving capacity of $c_i \ll c$ and a total system capacity of $c_{virt} = \sum_{i=1}^{s} c_i = \|\vec{c}\|_1$ with $\vec{c} = \{c_1, c_2, \dots, c_i, \dots, c_s\}$. The complete model now reads:

$$M(t)/G(t)/\|\vec{c}\|_1/0. \tag{3.7}$$

The instances do not have a static uptime. Instead, their life cycle is managed by a Virtual Machine Manager (VMM) and adjusted to the current load of the network. New tunnels are either placed on running instances or new ones are provisioned on demand. The VMM can have multiple optimization goals. A prominent example is the minimization of server instance and energy usage. Another set of example provisioning rules is discussed in the implementation of the model simulation in Section 3.3.

A target criterion could be to keep the blocking probability inside a certain target range. If the VMM decision rules are not carefully selected additional blocking could 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 makes the system more complex without providing real benefit, as failed tunnel requests are retransmitted by the network control plane or another attempt might also be made directly by the mobile device after a timeout.

To place incoming tunnel state on one of the available servers and manage the servers a load balancer is required. To ensure that the system 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 certain conditions are 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 should be considered in subsequent research.

Through this virtualized model, which suggests to use technologies from cloud computing in the network and replace specialized nodes with commodity hardware, network operators can scale the GGSN out instead of only up. Today, these network components are typically sold in a static and monolithic form and can not be easily extended with of-the-shelf hardware in order to accommodate to a changing environment. The system in this model can however be easily scaled out to additional low cost machines instead of completely replacing the existing GGSN with a more powerful version.

It is also entirely possible that the described single-arrival-process approaches might 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 models could be more appropriate in subsequent research. Still, the non-stationary Erlang loss model described here should be sufficient for basic core network control plane load estimations.

3.3 Load Model Queuing Simulation

As discussed, the solvability of a non-stationary Erlang loss system is very limited. To better tackle this, a simulative approach can be taken. Depending on the level of detail, different types of simulations are available.

Here, a queuing simulation is used to ascertain the blocking probability and tunnel serving slot utilization from the model using the fitted distributions from the trace.

3.3.1 *Queuing Simulation Implementation*

The queuing simulation is implemented on the basis of a DES. Instead of reproducing continuous time, this simulation is a series of discrete events. Time is advanced only at these events.

A queuing model can be easily represented in a DES. Each tunnel request arrival is modeled as a discrete event. When such an event occurs, three processes are executed. The first process draws a random number from a Pseudorandom Number Generator (PRNG) mapped to IAT exponential distribution to schedule the next arrival event. Secondly, the serving units are checked for any free units. If one is found, it will now be occupied. Else, this arrival will be marked as rejected and the third action skipped. This third process now determines the length of the tunnel using another PRNG adjusted to the serving time distribution to schedule the event in which the tunnel exits the system.

This model was implemented on the basis of version 3.0 of the *SimPy*[1] package, which is a Python DES framework that provides the basic event and scheduling infrastructure. On top of this a base GGSN class was constructed, managing the arrival of tunnel events and the scheduling of the service ending events. Specific classes for the traditional (i.e., monolithic) and virtualized (called "multiserver" in the code) nodes respectively exist.[2]

3.3.2 *Description and Design of the Individual Experiments*

To match the measurement data the simulation time is set to be 7 d in all simulation scenarios. The initial 60 min of each experiment are considered to be the transient phase and are afterwards excluded from the results. Ten replications of each scenario were performed. All depicted error bars show the 95 % confidence intervals across the experiments.

The first experiment was conducted to investigate the normalized baseline load a monolithic GGSN experiences using the presented model and the fits from Tables 3.2 and 3.3. Using this, an upper limit to the number of concurrent tunnels and the correlation to the blocking probability and tunnel rejection rate can be established. The effects of scaling up, improving the hardware capabilities of the single node, can thus be investigated afterwards.

1 https://simpy.readthedocs.org/

2 The implementation is also publicly available at https://github.com/fmetzger/ggsn-simulation/ as a reference.

Based on these results, the virtualization and scaling out effects in the virtualized, GGSN model are examined. In order to study the feasibility of this approach the performance indicators of the virtual GGSN are compared to the baseline established in the first experiment. To this end, the virtual GGSN is simulated in several configurations, which vary the number of instances and supported concurrent tunnels per instance.

In a final experiment the startup and shutdown duration of virtual instances and the life cycle management of these instances are additionally taken into account. Although the boot duration of modern OSs and Virtual Machines (VMs), especially on current hardware with flash storage, is significantly lower than it has been in the past, there is still a delay. This could cause further blocking if the load balancer does not account for this. But the more generously the balancer starts instances in advance the smaller the virtualization efficiency gain, especially the energy consumption, will be become. For this reason, the number of active instance is a relevant performance metric in the virtual GGSN model.

The experiment varies the boot delay and implements a very simple load balancer rule as baseline. The rule keeps at least one empty instance running in reserve at all times and deactivates instances, when two running instances are completely unused. As this is very generous, virtualization blocking should only occur in cases of instances limited to handling a low maximum number of tunnels or very rapid arrivals. Realistic provisioning rules can improve on this quite easily. But even this simplistic approach already serves to demonstrate potential benefits.

3.3.2.1 *GGSN Load, Capacity, and Scaling*

First, with the help of the IATs and duration of tunnels calculated in the dataset evaluation, the monolithic GGSN model is studied. While these passive measurement traces provided information on the frequency of new tunnel arrivals and the duration they remain active, no reliable information on the number of required supported concurrent tunnels for a given arrival rate could be deduced. This experiment evaluates arbitrary values for the GGSN tunnel capacity and determines the resulting blocking probability such that a reasonable value can be found, given desired limits on the blocking probability. This is a typical task in a dimensioning process.

Figure 3.17 studies this impact of the maximum supported number of concurrent tunnels n on the blocking probability p_B, where n is incrementally increased in steps of 100 tunnels from 0 to 5,500. As expected, the blocking probability decreases with the number of supported tunnels. An almost linear correlation can be observed in the larger part of the graph with a small convergence phase shortly before reaching $p_B = 0$. For the normalized inter-arrival rate no blocking is occurring if a capacity of 5,000 concurrent tunnels is allocated to the GGSN.

A similar picture is also evident in the number of tunnels served by this GGSN in the same scenario as shown in Figure 3.18. For the first half of the experiments the GGSN is loaded to its limit. Only when the capacity reaches 4,600 can the normalized arrival rate be fully served, which surmounts to about 3,820 tunnels on average in the system. Both results are stable across all simulation runs as the confidence intervals display. For the purpose of network dimensioning

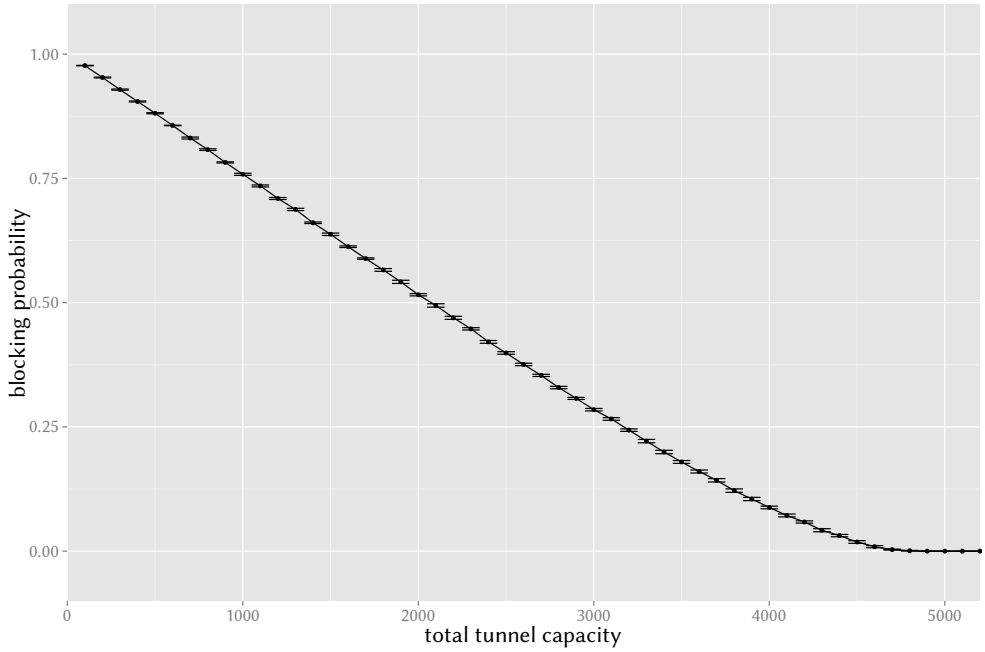


Figure 3.17: Impact of the number of supported parallel tunnels on the blocking probability for the traditional GGSN model.

the results can be easily scaled up from the normalized arrival rates to the actual ones in the network in question.

3.3.2.2 *Virtualization Impact and Gain*

A similar experiment can be set up for the virtual GGSN model. Learning from the monolithic model, these follow-up simulations can be tuned to the same total tunnel capacity in advance. The only difference is that the tunnel capacity is now spread out evenly between the virtual GGSN instances. The experiment tests different amounts for the total number of virtual instances, ranging from 1, which represents the monolithic architecture, up to 100 instances in steps of 10.

Figures 3.19 and 3.20 demonstrate the results in terms of p_B and concurrent tunnels served overlaid onto the base monolithic scenario's results. No large difference in the results can be seen and the virtualized GGSN model behaves no worse than a single large node model.

But the possible effects of an increased number of instances need to be investigated further. One goal in virtualization is the increase of energy efficiency. This can be achieved by having turned on just as many instances as needed and not more, thus scaling the system to its current load.

Therefore, Figure 3.21 takes a look at scenarios with nine different instance pools and varying tunnel capacities for each instance. Each setup is compared by the mean number of active instances during the one-week course. The bigger the capacity of each instance, the smaller the

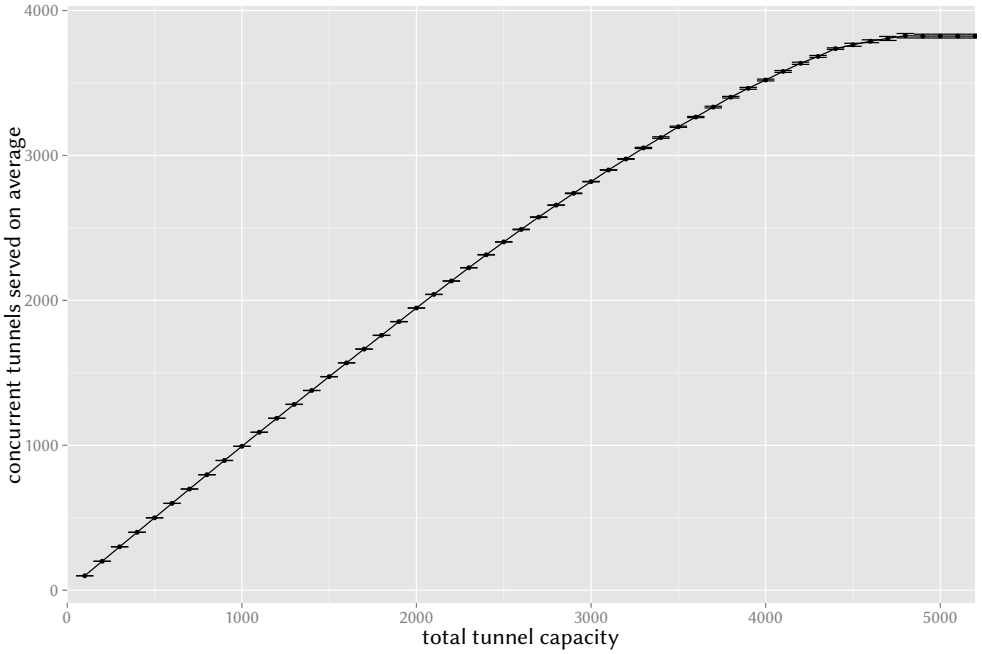


Figure 3.18: Mean number of tunnels concurrently served by the GGSN for incrementally increasing capacity.

number of instances required to be active. An actual GGSN, even a virtualized one, would need to be dimensioned in such a way to keep the total overhead low. It was already determined that, with the assumed normalized arrival rate, a capacity of 5,000 tunnels is sufficient in order to achieve a blocking probability close to zero. Keeping the setup at this minimum capacity and taking a look at the results in the figure, a good portion of the instances, usually around 20 %, can still be kept turned off.

To get into more detail, Figure 3.22 displays the distribution of the portion of time a specific number of instances was active. Depicted are four configurations that differ in their total number of instances and their tunnel capacity. The setup with 30 instances with 100 capacity was clearly overwhelmed with the arrival rate and all 30 instances were active over 70 % of the time. Only when the capacity was increased to 150 tunnels the virtualization benefits come into effect and more instances are able to sleep. Similar observations can be made in the 50 instance case. Here, the 100 tunnel scenario is already equipped to handle the tunnel arrival rate and can scale back its active instances quite well, below 40 instances half the time. The final configuration with a 150 tunnel capacity is clearly overdimensioned here with no more than 33 of the 50 instances ever being active.

Looking at these scenarios and additionally Figure 3.23 from a network dimensioning perspective, two distinct pathways to scale in the virtualized GGSN model are revealed. To reach the desired tunnel capacity either the number of instances or the instance's tunnel capacity can be increased. The latter represents the classical *scaling up*. But virtualization also opens up

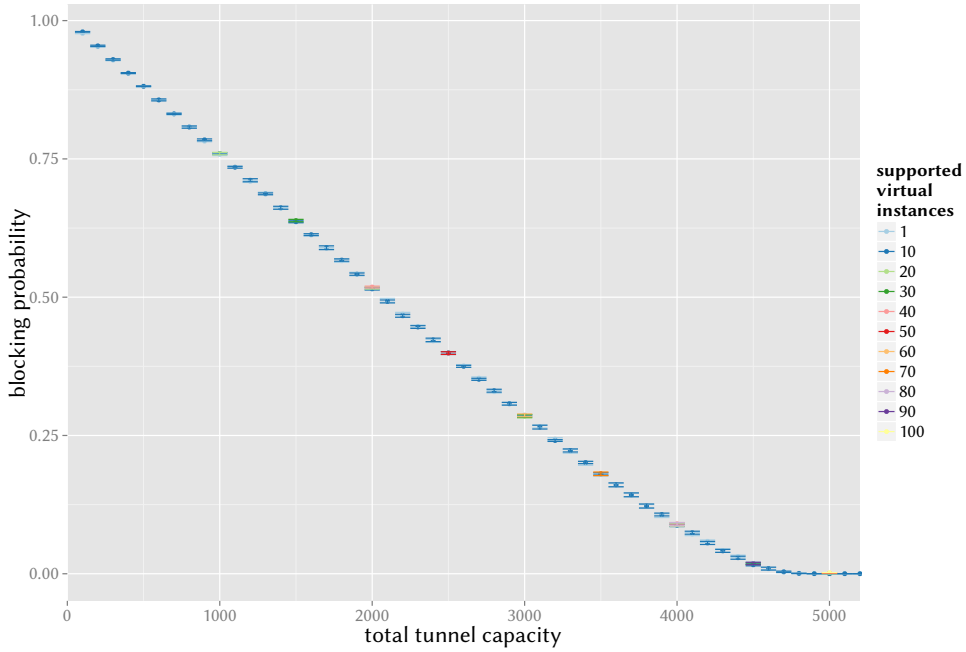


Figure 3.19: Comparison of the mean blocking probability of various virtual instance configurations. The horizontal axis depicts the aggregate capacity of all instances in the experiment.

the new path of *scaling out* by increasing the number of instances. Through this, scaling can become easier and cheaper as existing machines need not be replaced any more.

3.3.2.3 *Virtual Instance Life Cycle Management Impact*

A final aspect to be investigated in the simulation experiments is the potential increase of the blocking probability in virtualized scenarios when compared to the monolithic base. In theory, virtualization can incur additional overhead which would represent itself as an increase in p_B. In the given model the overhead can stem from the hypervisor and its scheduling and lifecycle management strategies in conjunction with the instances' boot delay.

The somewhat simplistic hypervisor strategy in this simulation was already discussed above and should give an upper limit on the impact on the blocking probability. This strategy is fixed, but the instance boot duration gets changed to analyze the impact. Values between 20 s and 5 min are considered and should reasonable represent real-life systems of a wide variety.

Figure 3.24 compares a number of instance and tunnel capacity scenarios on basis of their instance boot duration. In most scenarios there is almost no increase in the tunnel blocking probability. Only in cases with very many but small instances, where a lot of instance churn will occur, an increase can be noticed at higher boot durations.

Figure 3.25 investigates this increase in more detail and shows the blocking probability of one select scenario. Here, the system supports 5,000 tunnels in total with differing individual

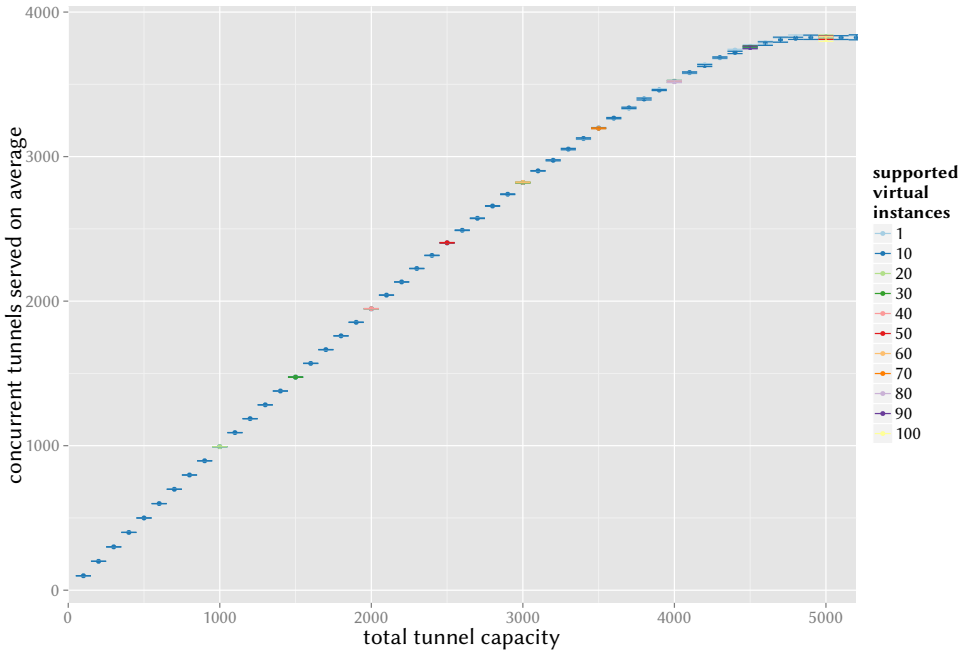


Figure 3.20: Comparison of the mean tunnel capacity usage of the individual virtual instance configurations.

instance capacity of 50, 150, and 500. In each case the start and stop down duration is changed between 1 min and 5 min. The increase in blocking probability in relation to both the instance capacity as well as the start duration can be easily observed.

This can be partially attributed to the assumed hypervisor and its simplistic scheduling and lifecycle management strategies in the simulation. If a low capacity instance with a long start time is activated, there is a high probability that the system will quickly expend its capacity again. A potential conclusion is that choosing larger instance capacities decreases the blocking probability at the cost of energy efficiency (because less instances can stay turned off).

Finally, Figure 3.26 shows two scenarios with 40 and 100 virtual GGSN instances respectively, ranging from 1,000 up to 5,000 served tunnels. For each scenario, the combined impact of different individual instance tunnel capacities as well as start up and shutdown time on blocking probability and mean resource utilization is studied. The first observation is that by increasing the number of instances, i.e., scaling out, the blocking probability can be decreased, while maintaining a relatively low mean resource utilization.

In addition to the previous effects, it can be noticed that a higher start up and shut down time causes a slight increase in blocking probability for instances with low tunnel capacity. Therefore, if smaller instances are to be used, for example due to price considerations, both the start up and shut down duration should be kept at a minimum. This could for example be achieved by using purely virtual instances in combination with fast storage.

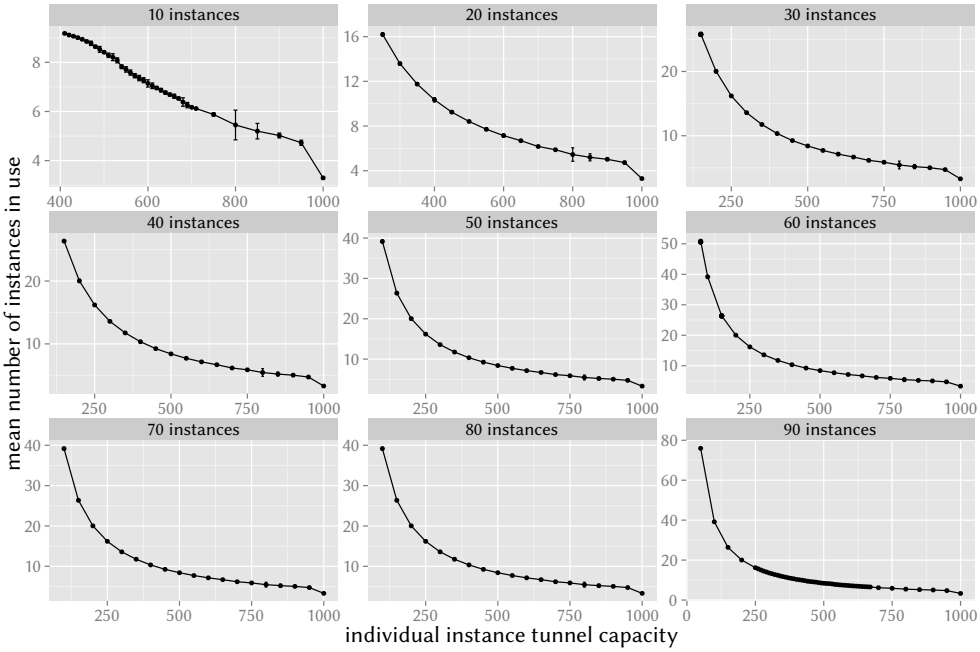


Figure 3.21: Mean instance usage of various virtualization configurations. A higher number of total instances results in a finer granularity of scaling and energy efficiency as more instances can be kept shut down.

3.3.2.4 *Significance and Effect Sizes*

In order to analyze the influence of the different model parameters on the resulting metrics a one-way Analysis of Variance (ANOVA) is performed. The effect size measures calculated here are the F-test, η^2, as well as ω^2 [Ell10; FMF12]. All are applied pairwise to each independent and derived variable combination. The results are depicted in Table 3.5. The two derived values and simulation metrics are the blocking probability and the mean tunnel usage.

Both of these metrics yield very similar results as they are also — by design — strongly related to each other. The F-ratio computed by the F-test and the corresponding significance level p indicate a large influence of the individual instance capacity on the metrics with a minor influence of the number of instances and no measurable impact of the start/stop duration. This is also confirmed by both η^2 and ω^2. Interestingly, only the compound variable, which describes the total tunnel capacity, i.e., the product of the individual instance tunnel capacity and the number of instances, is an almost perfect match in its variance to the derived metrics.

3.4 Core Network Evaluation Summary

The investigation of a week-long measurement trace recorded in an operational core network revealed some interesting signaling characteristics especially regarding the interdependency

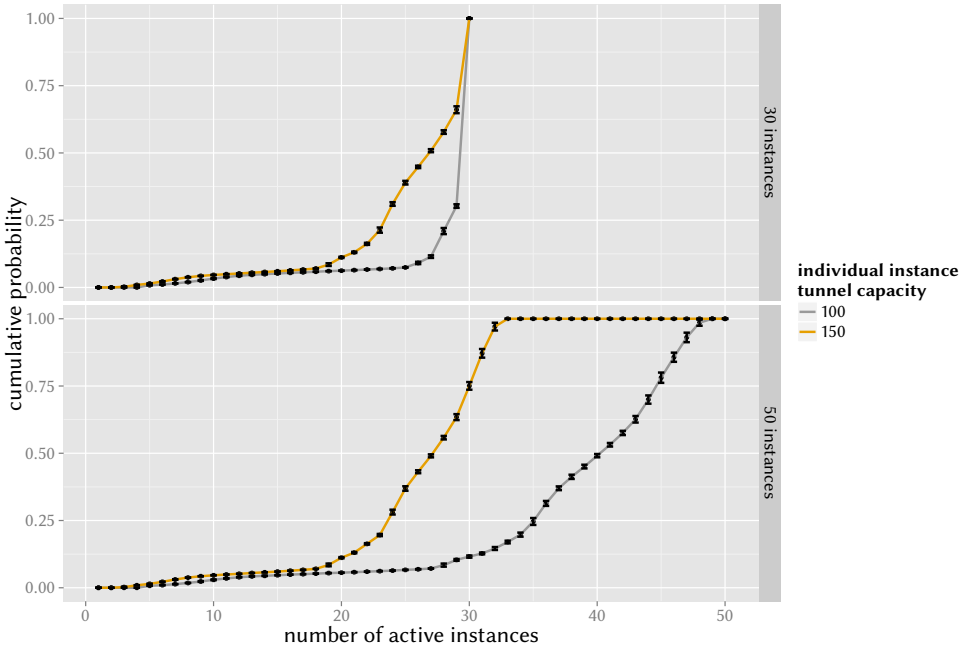


Figure 3.22: Impact of the maximum number of tunnels and number of instances on the number of active instances in the virtual GGSN model.

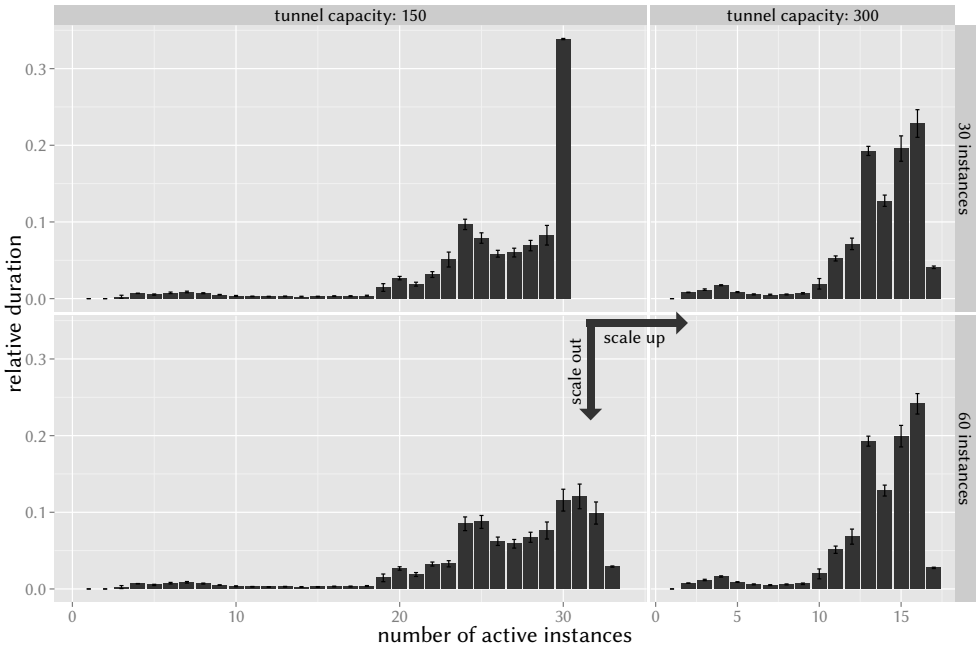


Figure 3.23: Resource usage from a select maximum instances and tunnel capacity combination, displaying the capability to scale up and out.

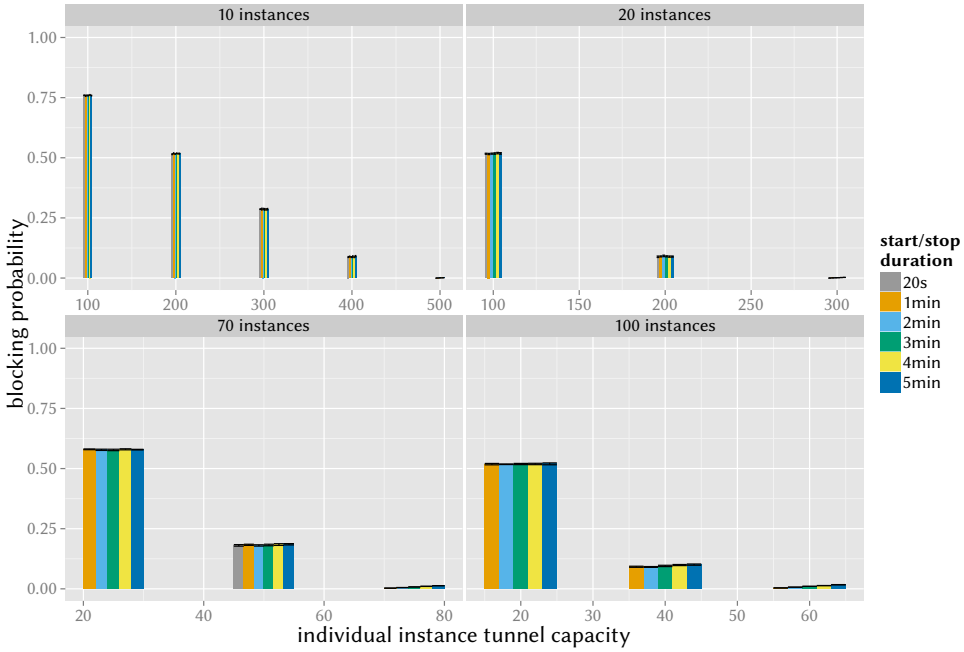


Figure 3.24: Influence of the boot and shutdown time on the blocking probability.

between user plane and control plane. Additionally, GTP tunnel properties were determined to be a worthwhile measure for control plane load at the GGSN, one of the central nodes in a 3G core network.

The investigation showed that the control plane is easily influenced by several device-based — as far as they can be distinguished in a core network trace — and time-of-day related features. The overall diurnal tunnel signaling load closely resembles the progression of the user plane. Most of the control plane's procedures are still triggered, either directly or indirectly, by user devices, of which the offered load is much smaller during night time. The trace evaluation also shows the currently dominating influence of smartphones compared to other devices types, even when looking at the control plane.

But this also means that sheer traffic volume is not a good measure to determine load, as the per-device traffic volume of a smartphone is rather low when compared to devices like pure 3G modems attached to a notebook. In this aspect, the findings also support the stories of signaling storms in mobile networks caused by applications regularly causing small amounts of network traffic. Each application interaction results in disproportionate amounts of signaling load being generated. Even worse, measures taken to improve the radio interface control plane such as Fast Dormancy could possibly have adverse effects to core signaling as they might increase the tunnel churn.

But the load investigation should not stop here. The presented approaches were just the ones that could be conducted with the available data. If one were to have access to a mobile network monitoring system or more detailed data records from such a system, it would open up many

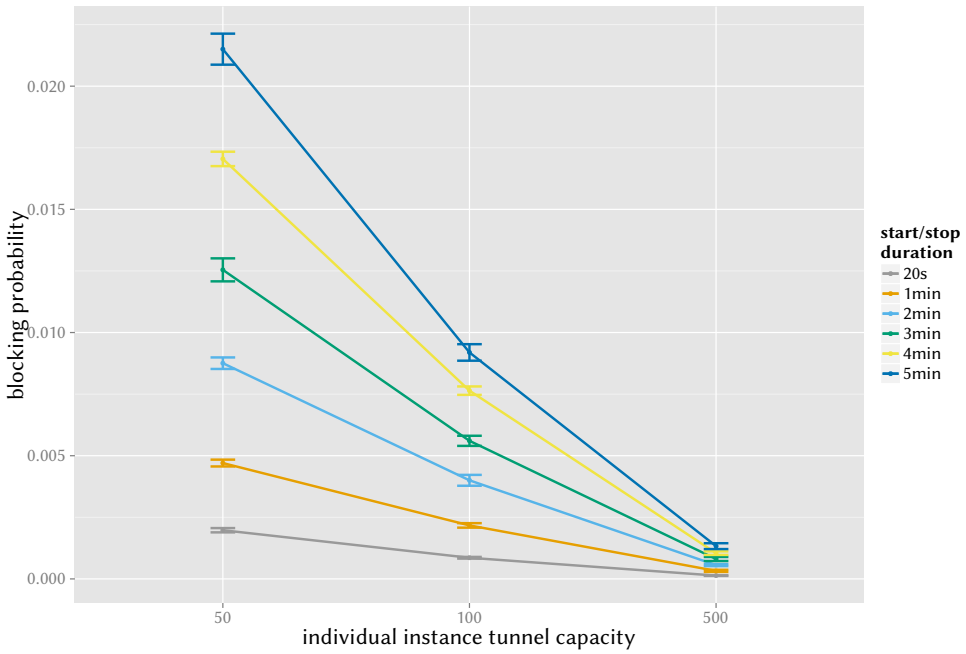


Figure 3.25: Influence of start up and shut down time on blocking probability with regard to different numbers of instances.

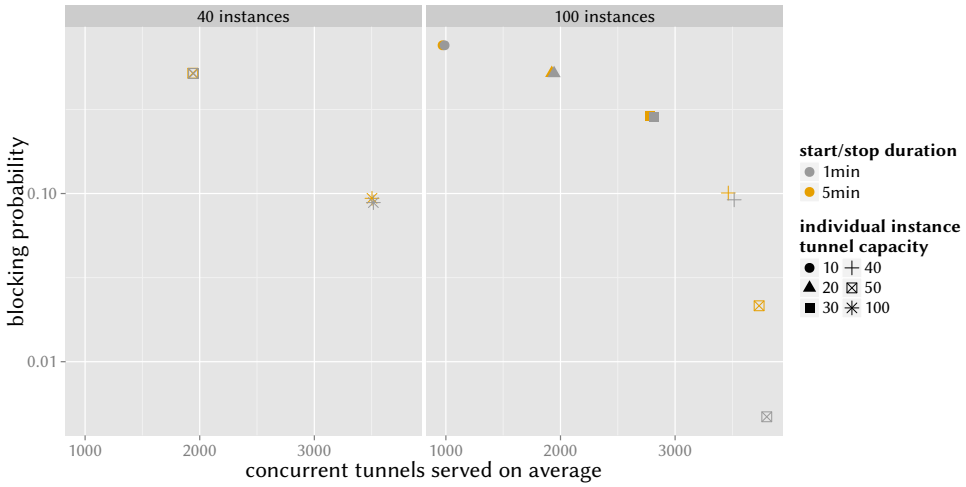


Figure 3.26: Trade-off between blocking probability and mean resource utilization with regard to maximum number of instances, instance tunnel capacity, and start and stop time.

Table 3.5: Effect sizes of the simulation parameters based on a one-way ANOVA.

	$F-ratio$	$p-value$	η^2	ω^2
Blocking probability				
Individual instance tunnel capacity	104	< 0.001	0.468	0.463
Number of instances	9.29	< 0.001	0.056	0.050
Start/stop duration	0.21	0.931	< 0.001	0.002
Total tunnel capacity	317257	< 0.001	0.999	0.999
Mean number of tunnels				
Individual instance tunnel capacity	105.7	< 0.001	0.472	0.467
Number of instances	9.39	< 0.001	0.056	0.050
Start/stop duration	0.25	0.912	< 0.001	0.002
Total tunnel capacity	365753	< 0.001	0.999	0.999

more angles in the investigation. For example, recording every individual signaling message with all IEs would give hard numbers on the direct signaling overhead, as could measurement probes located inside the network nodes report on the CPU and memory load in order to determine the control plane's processing overhead. A closer investigation of control plane load in relation to mobility behavior should also prove very interesting, as this is one of the central motifs in every mobile network.

Learning from this historical data, queuing theoretic models were created that can describe the control plane load in such networks. These models can be easily used in network dimensioning and planning processes by means of, e.g., stationary analyses. The novel baseline control plane load model presented here is a $M(t)/G(t)/c/0$ non-stationary Erlang loss model. When used in conjunction with parameters derived from the measurement traces it can easily be used for network dimensioning. To improve scaling in the future a further GGSN load model with features used in virtualization was also proposed.

Due to general solvability issues of non-stationary Erlang models the model is evaluated and validated using a queuing simulation in terms of their blocking and tunnel state probability as well as the overall resource utilization. The virtual model provided the added benefit of being more flexible in its scaling properties and energy efficiency. This might even lead to new GGSN-as-a-Service business models, removing the need to provide and operate large amounts of infrastructure for rare cases of peak load.

All of these properties serve to show the complexity of current mobile network systems even without running media streaming on top of it. Streaming in itself, while not being a real-time communication protocol, is relatively sensitive to timings and influences from lower network layers, which can make streaming over mobile networks rather problematic. The next chapters investigate these issues and methods to evaluate them more closely.

4

MEASURING AND MODELING RELIABLE VIDEO STREAMING

The Web would not have seen that big an increase in traffic in recent years if it were not for the tight integration of video streaming into every browser. Most forms of today's Web-based video delivery take advantage of HTTP and TCP to transport video. This is a completely different approach to what was used and was traditionally understood as video streaming in the past.

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. Each of them exhibits characteristic QoS properties. 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, through different radio receivers during mobility events, or when packets pass through excessively sized buffers.
- *Loss* occurs when data packets do not reach the target. One source of loss can be an imperfect physical medium that flips some bits in a data packet. The responsible higher protocol layer will recognize this and drop the packet or repeat the request.

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 while maintaining flexibility with regards to the actual streaming server implementation, the network, and also the protocol stack.

After defining an appropriate performance metric and researching various playback strategies, the model is implemented in a network emulation testbed. A measurement campaign is then conducted, testing the influences of various QoS settings and different playback strategies.

The work in this chapter was originally based on two publications, [F M+11] and [FRT12], but has been extended to include a broader categorization and modeling effort.

4.1 Streaming Definition and Classification

Before diving into the model some technical groundwork has to be laid. The protocols commonly used in the past and present are described and a classification scheme is set up. This is followed by a summary of other work related to this approach.

4.1.1 *Video Streaming Definition*

Any digitally stored **video** consists of a number of frames, organized into variable-sized groups of pictures, 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 dependency 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 much higher compression rate. To correctly display a frame, all previous frames in a group need to be present.

Streaming, or to be more precise video streaming, is the process of playing one part of a video while subsequent parts are 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.

4.1.2 *Streaming Classification*

Video streaming is a broad term covering a wide spectrum of applications as well as possible implementations. The following distinction criteria can serve to break down and classify this field.

4.1.2.1 *Video Source*

The first criterion is the source of the video transmission, with the two major sources being a stored file 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 watching a game that is being played right now).

4.1.2.2 *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 client's 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 by preparing and storing several encoding levels beforehand or by encoding the video on-the-fly for a specific target. While the latter approach has a much higher adaptation capability it cannot be precomputed and scales linearly with the number of number of clients (as opposed to a constant, one-off compute time effort for stored quality levels).

Adaptation also increases the necessary amount of control and information exchange. In a non-adaptive context, streaming would only require a single interaction to start the streaming while any single adaptation adds another set of interactions to change between quality levels.

4.1.2.3 *Location of Control*

Another matter is the location of control for a stream, with several possible ways to choose from. Between horizontal and vertical control can be distinguished.

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 located at the client side typically 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 called **pull-based** streaming.

Control can also be placed in the server. This is called a **push-based** approach as video data is pushed to the recipient. For this kind of control to work properly, state has to be kept imposing a certain memory overhead. The amount of state and processing of state can become 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 also 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. Push-based protocols usually employ an explicit state exchange.

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. Alternatively, a network could explicitly expose its QoS metrics for streaming applications to optimize themselves, or these application could send bandwidth reservation requests to the network.

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

4.1.2.4 *Reliability of the Underlying Transport Protocol*

A major differentiation can also be made based on the reliability of streaming. In the simplest case, streaming can act similar to a simple file download and just progressively download the video file in question while already starting to display the video contents. 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 at the cost 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 result in 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 UDP, which offers no reliability features like 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 by TCP, for example the adherence to an alloted or fair share bandwidth and congestion control. Otherwise, a high usage of this protocol could again lead to a scenario of congestive collapse as described in [RFC896].

Transport protocols that offer congestion control and no reliable delivery might be a desirable middle ground between these two extremes. Datagram Congestion Control Protocol (DCCP) [KHF06] is an example for such a compromise and might prove beneficial for the streaming process.

4.1.2.5 *Multiplexing of Delivery*

Finally, the number of targets of an individual 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 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.

4.1.3 *Survey of Protocols*

With these classification criteria at hand, an investigation for the motifs that are present in existing protocols can now be undertaken. The section largely describes RTP and compares it with HTTP-based approaches, including Dynamic Adaptive Streaming over HTTP (DASH), while also mentioning some other, proprietary, streaming protocols.

4.1.3.1 *RTP and Related Protocols*

RTP [RFC3550] is typically used in conjunction with its sister-protocol RTP Control Protocol (RTCP) and often also employs Real Time Streaming Protocol (RTSP) [RFC2326]. According

to literature, they are the classic approach to video streaming[1]. The protocol suite employs a *push-based approach* with the RTP server application in full control of the streaming process. Control and information exchange is conducted 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. These 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. Every 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 [RFC3640; RFC6184; RFC6416], 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 sub-structure inside the packet. For example, if dealing with an MPEG-4 Elementary Stream (ES), the payload may contain one or more Access Units (AUs).

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 and are scaled in such a way that the bandwidth should not exceed 5 % of the actual stream's bandwidth. The reports will include statistics related to lost packets as well as the packet delay and its 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 and RTCP on their own provide no means to discover, initiate, and control the streaming process and have to rely on additional protocols. RTSP is one 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.

1 For example, refer to [KRA08, p. 589ff] or [PD07, p. 426ff]

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) [RFC4604] without the need for RTSP. However, all intermediary routers have to support this mode. Therefore, it is usually only seen in closed networks where the whole network infrastructure is owned by a single organization. For example, this scheme is often employed by Internet Protocol television (IPTV) providers in their access network.

RTP is also used extensively in conjunction with other protocol suites, including Session Description Protocol (SDP) [RFC2327] and Session Announcement Protocol (SAP) [RFC2974] for stream discovery or in realtime communication protocols such as SIP [RFC3261] and Extensible Messaging and Presence Protocol (XMPP) [RFC6120; RFC6121] with the Jingle multimedia session control extension[2].

The prevalent presence of middleboxes in the Internet also poses issues with RTP's requirement of several simultaneous UDP streams. Network Address Translation (NAT) nodes are especially problematic because of the difficulty to forward incoming UDP packets to the destined host. This can be partially circumvented by using NAT traversal techniques like Session Traversal Utilities for NAT (STUN) [RFC5389] or Interactive Connectivity Establishment (ICE) [RFC5245], sometimes though unreliably.

4.1.3.2 *HTTP Streaming*

When compared to RTP, HTTP streaming represents a much less specialized approach by only reusing existing general purpose protocols. The HTTP/1.1 [RFC2616] application layer protocol is the basis of the Web and is a request/response protocol mainly to retrieve and pull files from and to a remote location.[3] The protocol is stateless for the server, requests are fully independent of each other and will be responded to only with the provided metadata.[4] This holds true even when more than one request is sent over the same TCP connection, which can be achieved using 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.

HTTP can also be easily exploited for video streaming. The file to be retrieved should of course contain video and all frames have to be stored sequentially. If there are separate streams present in file, most commonly at least video and audio, they must be interwoven. Video metadata necessary for the start of the playback process, which typically includes sub-stream information and codec parameters, needs to be positioned 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

2 The draft standard is available at `http://xmpp.org/extensions/xep-0166.html`.

3 In June of 2014 the original RFC has been obsoleted and replaced with a new set of specifications in preparation of HTTP/2. The specifications are: [RFC7230; RFC7231; RFC7232; RFC7233; RFC7234; RFC7235; RFC7236; RFC7237; RFC7238; RFC7239]

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

separate files, potentially simplifying the file structure. However, this increases the complexity of synchronizing both streams at the video player.[5]

The actual streaming is controlled completely by the player application at the client. This player simply has to issue a HTTP 'GET'-request for a video file stored at the 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 to avoid to run out of buffered data at any point during playback. Approaches to this task will be explained in detail in Section 4.3. HTTP also allows so-called range requests, which allow 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. Otherwise, the range requests have to be guessed.

Reliability and Adaptivity

HTTP uses TCP as transport protocol, which has implications to HTTP streaming and distinguishes itself significantly 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 for the lost packet to be retransmitted, causing Head-of-line (HOL) blocking. On links with high loss the transmission could 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. Reliability for video streaming means that the whole video (or at least the current segment) has to be fully played in sequence and no implicit quality adjustments based on the current bandwidth are possible.

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 bottleneck link bandwidth. 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

5 Stream synchronization issues are for example present in RTP.

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.

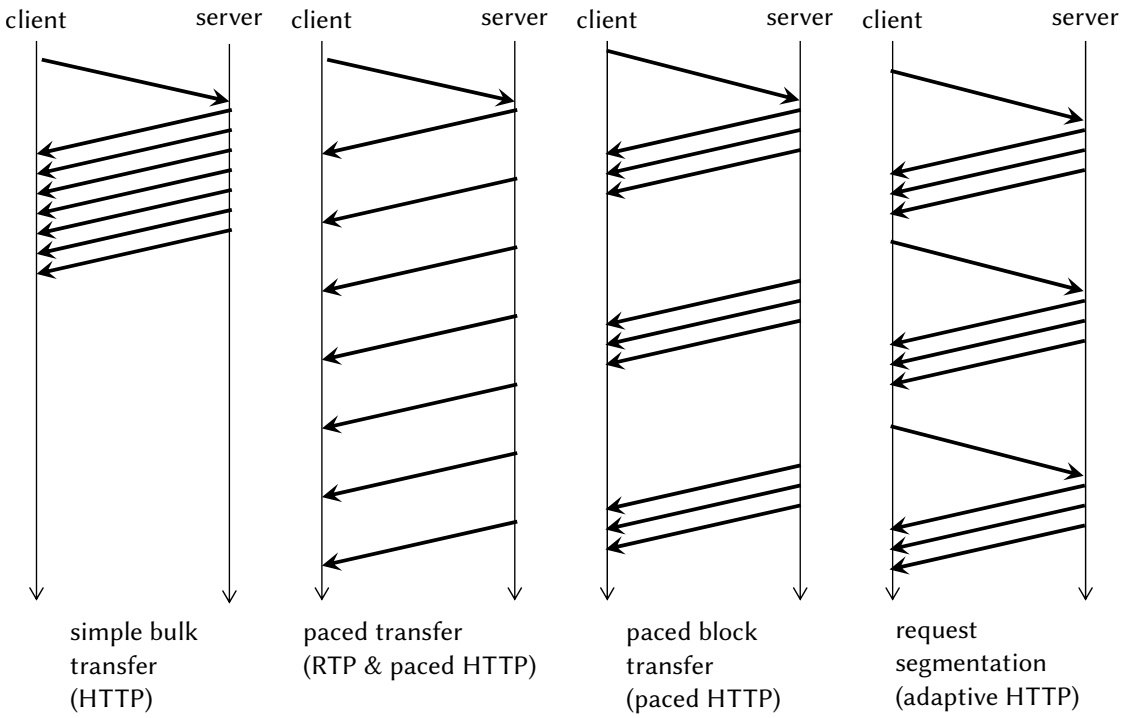


Figure 4.1: Comparison of several possible streaming transmission modes depicting the timing of the sent packets (source: [Ma+11]).

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 4.1 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 evenly-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 as observed in [AN11]. 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 [F M+11].

These flow control mechanisms only adapt the transmission rate to the stream's bit rate but not vice versa. Having access to the video in different bitrates may be desirable for many use cases, especially for reacting to changing network conditions. Take a vertical handover from a high bandwidth WiFi network 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. A longer overview is for example given in [Ma+11] or [BAB11a].

Several formal **adaptive streaming** protocols have been standardized or 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/IEC [MPE12] and 3GPP [3GP13l] standard. Herein, all video segments are gathered in an alternative XML-based presentation scheme. Using stored video for HTTP streaming is more appropriate due to the file-based nature of HTTP. But the streaming protocols have also been successfully employed for live content, for example 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. Proxies are usually not employed for video streaming but can potentially alter and adapt the stream to the needs of certain clients. However, an adaptation inside the network usually requires much greater efforts with less effects than at the endpoints.

Through 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 Domain Name System (DNS). Caches are usually placed in close vicinity of potential receivers. This creates a so-called Content Distribution Network (CDN) avoiding both long transmission paths through the Internet and the single server bottleneck. 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 actual multicast-enabled streaming would not save any bandwidth there. For an exemplary investigation of YouTube's CDN structure refer to [Raf+11].

Currently, HTTP is in the process of receiving major remodeling with the efforts of WebSocket[6] [RFC6455], SPDY [BP13; BP12], and ultimately also HTTP/2 [BPT14]. All three improve the flow multiplexing capabilities of HTTP and allow the server to initiate transmissions on its own. This enables more control possibilities for the server and can improve any segment-based

6 http://www.websocket.org/

adaption scheme as these segments do not need to be requested anymore but could just be pushed by the server at a convenient time.

4.1.3.3 *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 3GPP specification for multicasting multimedia traffic in the mobile network architecture. The explicit control structure of protocol suites like MBMS and also 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 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, 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[7] and Zattoo[8]. Table 4.1 attempts to summarize all the protocols of interest to this research and apply the proposed classification criteria to them.

Table 4.1: Streaming 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)	using IGMP
simple HTTP	in-band, streaming application	client-side	reliable (TCP)	stored, not live	none	emulated through CDN
adaptive HTTP (e.g. DASH)	in-band, streaming application	client-side	reliable (TCP)	stored and near-live	client-side with file segmentation	emulated through CDN

7 https://www.tribler.org/trac
8 http://zattoo.com/int/

4.2 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. This section attempts to summarize some of the more closely related research approaches with today's reliable streaming mechanisms as a central theme.

The chapter's focus lies on reliable HTTP streaming to which [BAB11a; BAB11b] and [Ma+11] gave an introduction and overview the mechanics involved in streaming, e.g., flow control mechanisms in the video delivery. Akhshabi et al. [ABD11] took a look at real world streaming implementations and conducted comparative measurements. Initial experiments evaluated 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 showed detailed characteristics of the served videos, amongst others file sizes, durations, and bitrates, and revealed a daily number of video requests. Adhikari et al. [AJZ10] collected data from points of presence of one Internet Service Provider (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 exhibited large differences to the old one. For example, load balancing and content distribution now exclusively use Google's network and is no longer based on Akamai's infrastructure.

Mori et al. described in [Mor+10] distinctive attributes of video traffic flows originating from YouTube's new and current setup, while Torres et al. [Tor+11] focused 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 an interdomain traffic study. The amount of Web and video traffic was seen to be on the rise, both is to be expected through the presence of large video distribution Web sites.

The authors of [Wan+03] proposed 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]. However, the loss-hiding properties of reliable streaming makes this study only somewhat applicable to HTTP streaming. In [Kaw+10], the authors presented 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 it rather focused on the codec performance instead.

Video quality, or so-called QoE metrics, help in determining the quality of the streaming process and of models resembling the process. The metrics can be either subjective or objective and are further discussed in Section 4.3.1. A few select publications are already presented here.

A metric coined "application comfort" was defined and applied to YouTube videos in [Sta+10] to monitor live network conditions in realtime. It is geared towards a very specific implementation of streaming, whereas the measurement methodology presented here is more generic.

While YouTube's horizontal location of control is originated at the endpoints, some third-party control unit could also be placed in an access network, manipulating YouTube streams in an attempt 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 ISPs to check their network quality and estimate the quality customers are achieving.

A publication by Hoßfeld et al. [Hoß+11] identifies QoE influencing factors on YouTube streams through subjective Mean Opinion Scores (MOSs) collected by a "crowdsourcing" method. The number of stalling events was revealed to be the factor with the highest impact on the QoE. This idea was furthered by [Hoß+12] with a comparison between the initial delay of a stream's start and the number of interruptions. Stalling resulted in a much lower MOS. Observations performed in [Ket+10] on the Android platform were also showing that stalling events can result in a large drop in MOS.

The authors of [MCC11] measured QoE effects of HTTP video streaming in a controlled test setup and conclude that degraded network QoS increases stalling frequency and decreases the MOS. Gustafsson et al. [GHP08] investigated the loss in perceived streaming quality and established a parametric objective opinion model. [SHR12] presents another QoE model that attempts to estimate the quality of adaptive streaming with a neural network trained through subjective tests.

Adaptive streaming has also been a topic of intense research. [DMP11] investigated quality adaptation techniques and proposes a feedback mechanism for quality control. Curiously 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 and found video streaming traffic as ubiquitous as in wired networks. The authors of [Hua+12] concluded 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 was developed in [Rao+11] through the evaluation of active measurements comparing different streaming strategies. And finally, in a survey of several streaming protocols in [MHM10] the overhead on the transmission of each of them was investigated and compared based on an analytical approach.

4.3 Streaming Modeling

As stated, the goal of this chapter is to model the measuring process and reliable streaming. This section will introduce the proposed reliable streaming measurement model and with it several reference playback strategies that cover many of the potential aspects of streaming. It is intended for an easy comparison between different protocol variants and to evaluate all strategies and network influences in one framework. Because any kind of evaluation requires metrics, and reliable streaming has special requirements in this regard, appropriate metrics are researched first.

4.3.1 *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 denominated MOS. Through the human element, conducting a subjective assessment is very time and resource consuming, but it can also serve as the best indicator of the QoE.

This is where objective video quality assessments come into play. Modern objective models attempt to recreate the features of human visual perception and psychological model and are usually calibrated by subjective assessments. Most objective models operate on a full reference approach, meaning that they directly compare the original reference video to the resulting video after being encoded 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:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2 \tag{4.1}$$

and

$$PSNR = 10\log_{10}\frac{L^2}{MSE}, \tag{4.2}$$

where N denotes the number of pixels in a frame, and x and y the individual pixels from the reference and output frame respectively. The quantity L denotes the maximum value of a pixel. Usually L is from the range $[0, 255]$, corresponding to a bit depth of 8 bit. The image is either grayscale, or, if it has color information, each color channel is calculated separately. [WSB03] offers more information on these metrics.

Image quality models can by nature only test for spatial distortions of single images. This includes artifacts like general blockiness or blurriness, noise, or reduced resolution. Quality assessments specifically dedicated to video material can additionally take temporal metrics into account, for examples anomalies in the frame rate. Such models are also being researched

and standardized by the ITU and Video Quality Experts Group (VQEG) for example in [ITU04; ITU08b; ITU08c]. Often, a network's QoS parameters can also be directly mapped onto an objective QoE metric as described in [FHT10].

Metrics dedicated solely to streaming quality measurements should restrict themselves to measure only degradations that occur during the streaming process and not the initial encoding process. Only a specific subset of quality alterations apply here. For example, lost or late packets can cause missing blocks in a frame or frames to be skipped completely.

Initial thoughts concerning QoE in IPTV systems — which represents an unreliable streaming solution — QoE have been given in [ITU08a] and the influence of packet delay variations on playback buffers is investigated in [RFC3393]. The Media Delivery Index (MDI) [RFC4445] 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

$$VB = r_{\text{rcv}} - r_{\text{drain}} \tag{4.3}$$

$$DF_i = \frac{\max(VB) - \min(VB)}{r_{\text{drain}}} \tag{4.4}$$

with r_{rcv} and r_{drain} denoting the buffer's receiving and draining rates.

Reliable streaming is even less likely to degrade the image quality of a video stream. Packets can not arrive out of order at the application and loss is fully concealed by TCP. This means that the transmitted and the played video are always identical and no image reference models are necessary here. The only thing that can still happen, is that portions of the video data arrive too late to be played out at their intended point in time. Reliable streaming quality assessment metrics 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. More complex metrics could also keep track of the IAT of stalling events.
- For adaptive streaming, the characteristics of the quality levels the video was played in. This includes the frequency of quality switching events and the duration of each level.

Few attempts for concise reliable streaming metrics that cover all mentioned properties have been made to date. 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 $I_p = uv$ is simply based on the number of stalls N and the average stall duration v. Very similarly, an ITU recommendation [ITU13] defines a rebuffering artifact value as

$$v = 1 - \left(\frac{d_s}{d_v + d_s} \right)^{0.0737}, \tag{4.5}$$

with the total stalling duration d_s and the video duration d_v. The model lacks to capture other stalling characteristics and quality levels. A final metric by Hoßfeld et al. [Hoß+13] remedies this and incorporates both the length L as well as the number N of stalls in the computation of the MOS. It states

$$f(L,N) = 3.50e^{-(0.15L+0.19)N} + 1.50 \text{ for } L \in \mathbb{R}^+, N \in \mathbb{N}. \tag{4.6}$$

The equation was derived from user studies on perceived YouTube video quality. Therefore, it can serve as a QoE metric for the non-adaptive measurement model introduced in the following section. Regarding adaptive streaming, work conducted in [Seu+13] gives an overview over some initial attempts of metrics that include temporal aspects of the image quality. But none of them includes all necessary aspects to assess reliable adaptive streaming.

The streaming measurement models presented in the following section derive only the basic properties, including the stalling and quality level characteristics, from the experiment and could be used for any of the above metrics. For non-adaptive reliable streaming it is recommended to use the model defined in Equation 4.6. The results of the measurement series in Section 4.4 will also be presented using this equation. Surpassing this model or defining an entirely new adaptive metric would require extensive user studies and is beyond the scope of this work.

4.3.2 *Measurement and Playback Model*

Parts of the model presentation were previously published in [BFT12], [F M+11], and [FRT12]. 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 the next portion of the video to be displayed. From the perspective of a player application, a video consists of a sequence of atomic units, namely video frames and audio samples, which need to be played back at certain points in time, predetermined through the video's frame rate.

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 and subject to the network QoS. The process can be subsumed as:

$$buffer(t) = \sum_{0}^{t} \text{data}_{\text{received}} - \sum_{0}^{t} \text{data}_{\text{played}}$$

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 must stop and stalling occurs.

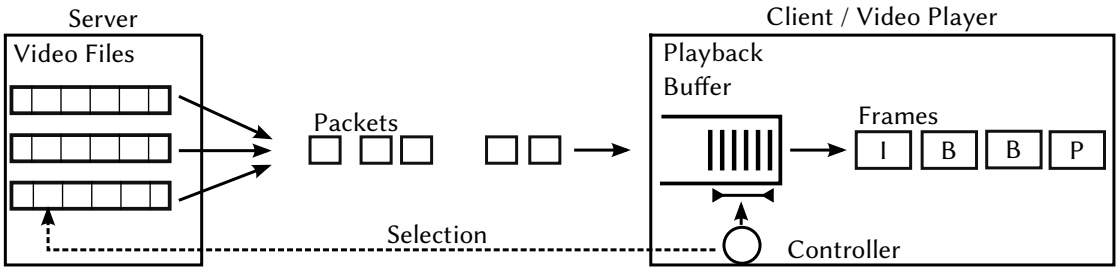


Figure 4.2: Reliable streaming playback model based on buffer control.

Figure 4.2 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 four degrees of freedom, which are all governed by the controller and together are coined **playback strategies**. These degrees of freedom are are:

- 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.
- If the video is segmented the segment retrieval rate is also a factor. Individual segments can be requested at a higher rate to fill the buffer more quickly. With this the client can react to changes in network conditions.
- Selection of the video or segment quality level with a video bitrate chosen according to the current network throughput. This is only applicable for adaptive streaming of course.

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 strategy. Under insufficient network conditions the playback strategy has to decide which property is less likely to negatively impact the user. Therefore, either the frequency or the length of stalling events will have to be adjusted. The more frequent the stalls are, the shorter they will be. If the strategy produces longer stalling events, 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. To keep the buffer level high, an adaptive streaming client can also reduce the stream's quality level. But it is still an open question if playing at low quality without interruption gives a better QoE than high quality with interruptions.

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

4.3.2.1 *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. Both the total stalling time and the required buffer space are minimized. Moreover, the strategy results in an upper limit for the number of stalls occurring[9]. Therefore, it can act as a baseline reference to assess the performance of other strategies.

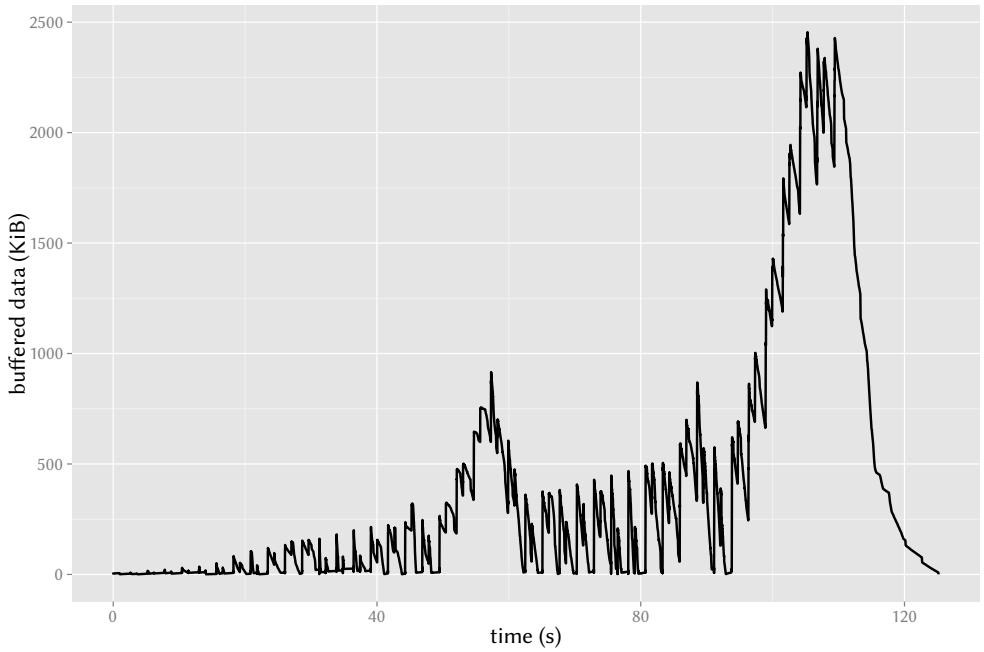


Figure 4.3: Buffer fill level with null strategy; 33 s total stalling.

Figure 4.3 depicts an exemplary time series diagram of the contents of a video buffer using this strategy. The transmission rate was only slightly above the stream's video bit rate. The buffer frequently drops down to zero forcing a short stall. According to the related work in [Hoß+11], on the QoE impact of stalling frequency in comparison to the length of stalls, this is the worst possible scenario for a person watching the stream.

9 As a video frame is atomic, no other model could possibly need to stop the playback due to buffer drains more often.

4.3.2.2 *Threshold Strategies*

Instead of instantly restarting playback a 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 or 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 wallclock time — this can also be seen as threshold — and to start playback after that period regardless of the volume of the buffer. Additionally, the threshold could either be set to a constant value or dynamically chosen according to the expected network QoS.

Besides this single-threshold strategy, multi-threshold strategies 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.

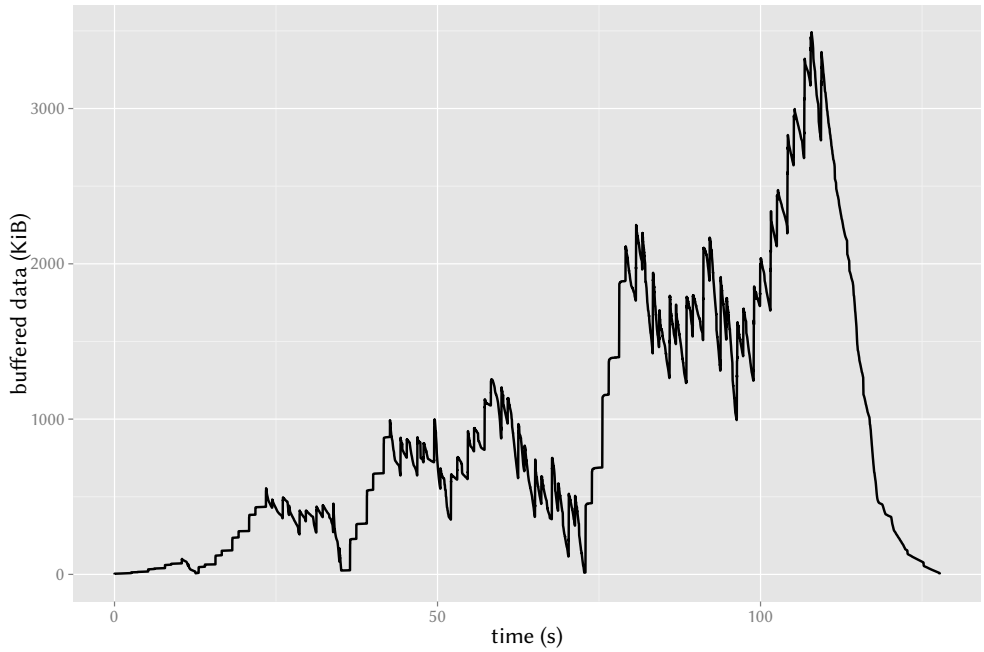


Figure 4.4: Sample buffer fill level for a 5 s buffered video duration threshold strategy with an additional 2 s initial threshold; 34 s total stalling.

An example buffer diagram is displayed in Figure 4.4. 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.

4.3.2.3 *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 temporal 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.

4.3.2.4 *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. An exact implementation of this strategy would require precise information — so-called global knowledge — on future events and therefore it can only be conducted in a theoretical offline manner. Instead of global knowledge, heuristics can instead attempt to approximate an expected future state.

A very simple predictive approach is to prolong the initial delay to the point that no intermediate buffer underrun and thus no further 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 4.5 depicts the time series of a sample implementation of this delayed playback predictive strategy with all necessary stalling occurring upfront.

4.3.2.5 *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 finer-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.

A practical attempt at defining an adaptive strategy is made in Section 6.2 for the mobile streaming testbed.

4.3.2.6 *Real World Implementation Examples*

Actual streaming player implementations often do not implement just one of these strategies, but rather combine ideas from several. Herein, thresholds are often set arbitrarily through best

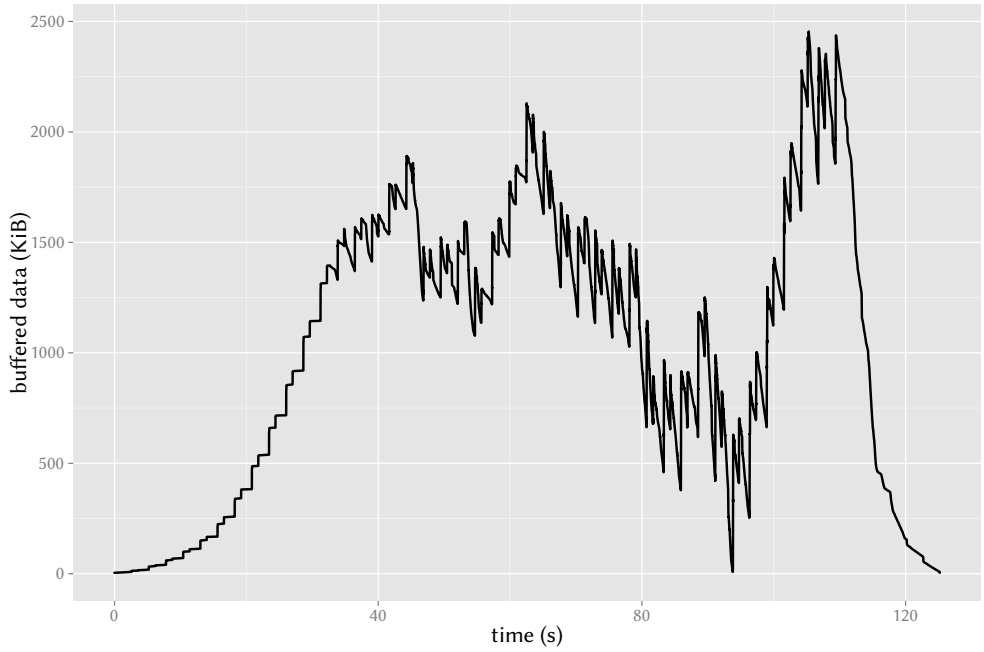


Figure 4.5: Sample Buffer fill level for the delayed playback predictive strategy, 33 s total stalling.

practices which are not empirically evaluated. Also, often a trade-off between user perceived quality and the resulting server load is made as a business decision that can further impact the streaming quality. In general, every streaming service essentially 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 4.4. 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.

Furthermore, YouTube employs a proprietary server-side pacing mechanism outside of the control of the streaming player. This is hinted at in the encoding of the URLs of the video files and enforced by the video file cache server. Some of those (not user-changeable) parameters are

Table 4.2: 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.

described in Table 4.2. The pacing was in effect for all videos below high definition resolution, but has since been extended to include all video files.

The throttling method, which was 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 1.25 times the video bitrate. The rate limit is not constant, instead an 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 4.6. 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's 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.

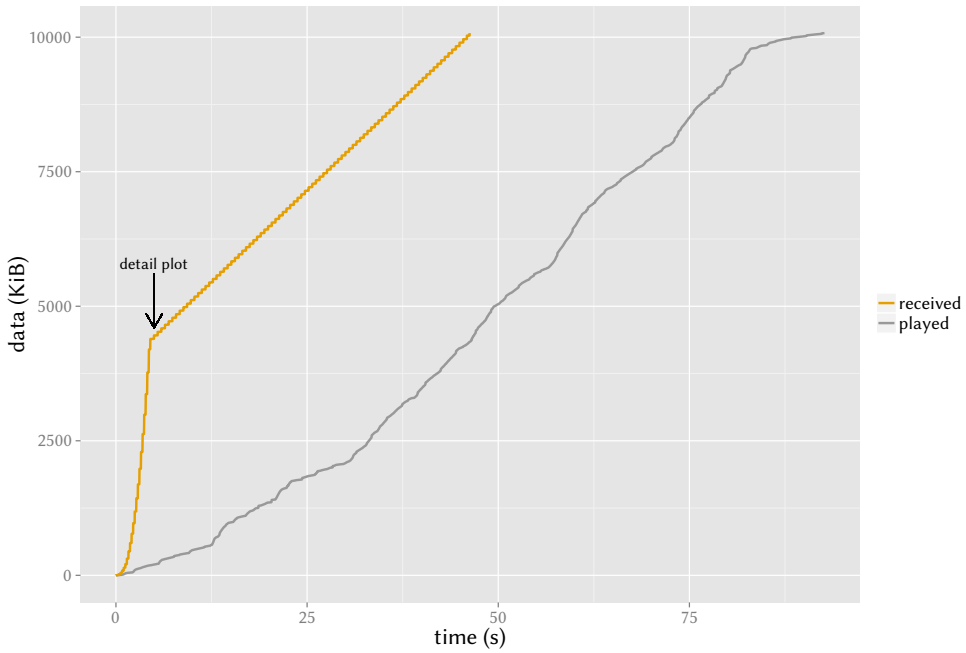
Since October 2013 YouTube is serving its HTML5 videos using DASH, albeit without its adaptive properties. Additionally, in mid 2014, this HTML player has now become the default player for all capable browsers, effectively making the Flash player obsolete. Some initial investigations into YouTube's DASH traffic were conducted in [ASO14].

Firefox's HTML5 Player Strategy

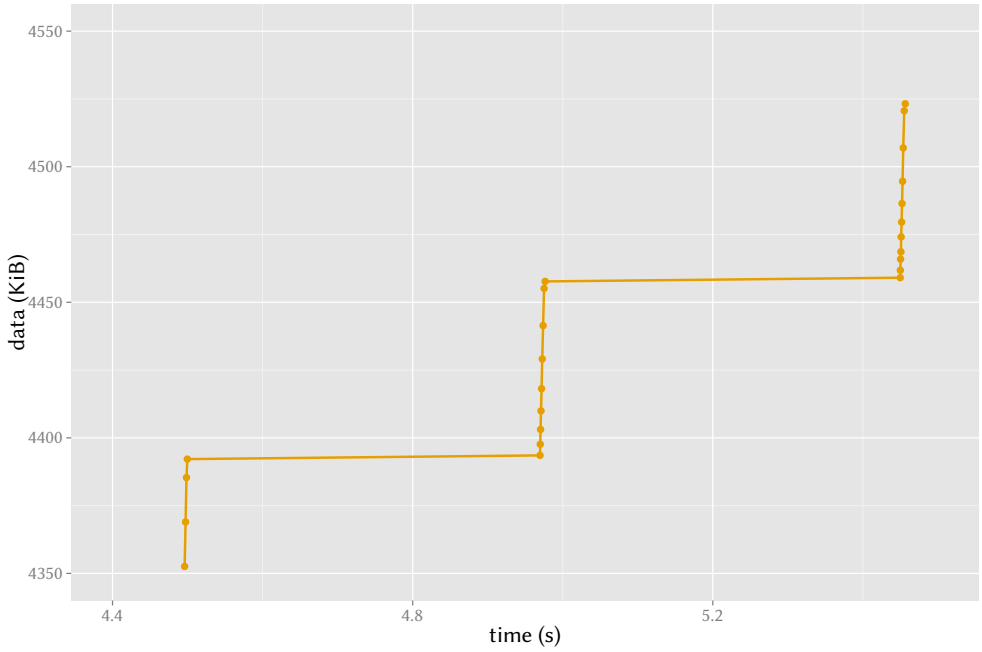
Video streaming can be directly conducted with the Web browser, through the use of a HTML5 canvas element. The World Wide Web Consortium (W3C) specifies the default technical process of HTML5 video streaming[10] and essentially suggests a predictive strategy. Herein, the Web browser should estimate and 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.

10 http://www.w3.org/TR/html5/embedded-content-0.html#media-elements



(a) Overall graph.



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

Figure 4.6: Comparison of downloaded and consumed data volume revealing the pacing mechanism used by YouTube.

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

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 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.

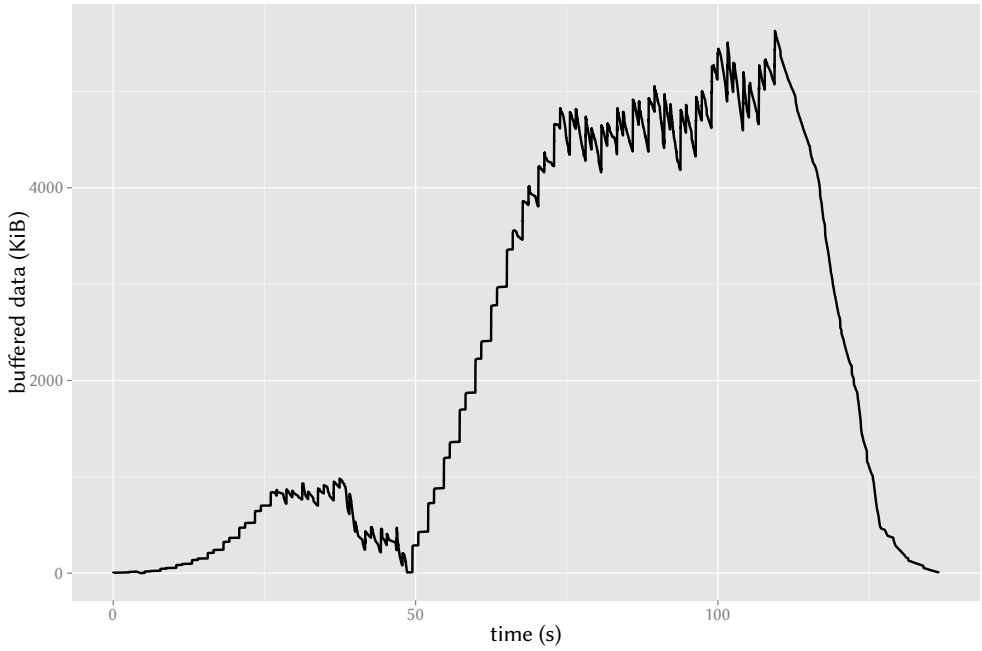


Figure 4.7: Sample buffer fill level for the Firefox 4 strategy, 44 s total stalling.

This approach is quite conservative and trades off long stalling periods for fewer stalls. The test case for the model is shown in Figure 4.7. 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 scarce amounts of memory, e.g., mobile phones. A large buffer could, on the other hand, also increase the chance of continuous playback in such scenarios with insufficient 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 protocols with open specifications. These typically only define the transport and media format but explicitly do not specify the players' behavior. DASH is one of these cases.

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].

4.4 Measurements

With the buffer-based playback model and strategies 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 every time with slight variations in execution, choice of playback strategies, and corresponding parameters. 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 computers. Moreover, these players are built with user interaction and not automation in mind, hampering efforts of directly measuring the outcome. This can still be achieved through extensive workarounds, but those must be tailored to every individual player application.

The approach presented here avoids this hassle and provides a concise way to test any conceivable playback strategy in one single test setup.

4.4.1 *Progressive Streaming Measurement Framework*

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

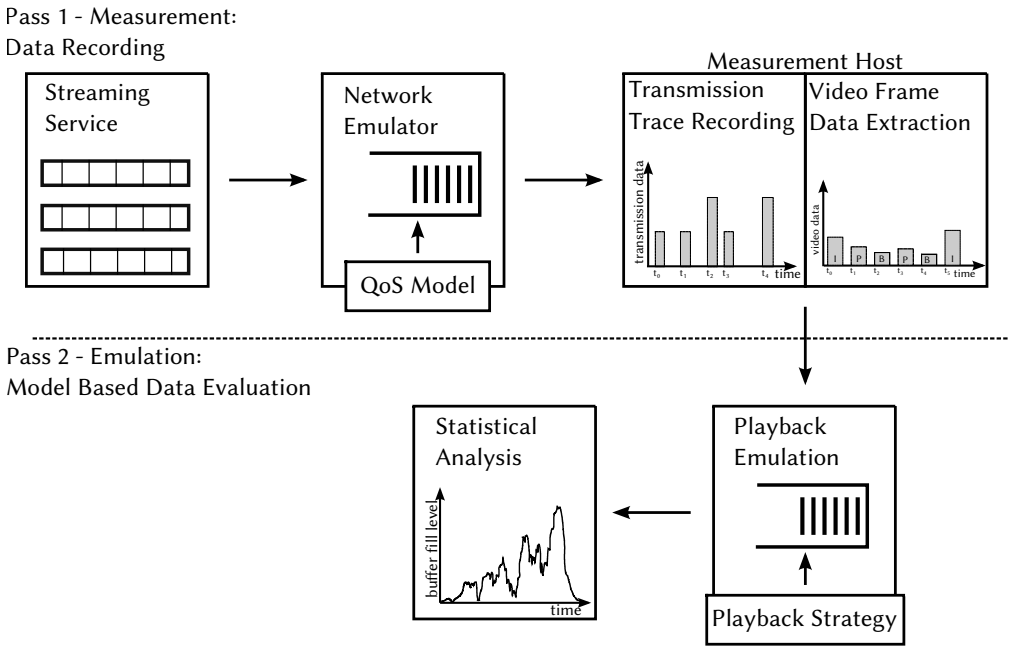


Figure 4.8: Overview of the measurement framework for progressive streaming playback strategies.

Figure 4.8 depicts the usage of the framework for a streaming evaluation testbed. In phase one the actual transmission of the stream is conducted and recorded as a packet level network trace. These traces should at least consist of the size and timestamp of every incoming packet.

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 further 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 such as the 3G mobile network architecture.

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. Of special note for this work are the previously discussed mobile networks, which encapsulate the user traffic into tunnels and exhibit complex control plane interactions that can influence the streaming process.

Additionally the received video file is decoded yielding a trace of all video frame sizes and playback timestamps. All data gained in the process is stored as a basis for the second phase. More detailed traces can additionally be used to scrutinize other layers of the connection, e.g., the dynamics of TCP receive window size.

In the second pass, both the network as well as the video data are then used to feed the actual reliable streaming playback model described before. This is 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 non-adaptive reliable streaming strategies can be tested on the same trace set. In the simplest form of 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 a fast and efficient comparison of various non-feedback protocols which are all subjected 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 closed source streaming clients can be a difficult and time consuming reverse-engineering process.

4.4.2 *Adaptive Streaming Measurement Framework*

Up to this point, the measurement framework is only suitable for simple reliable streaming strategies but neglects any adaptive strategy. This second iteration of the framework modifies the base framework and allows for the testing of adaptive playback strategies. However, to achieve this, the advantageous two-phase setup cannot be employed anymore.

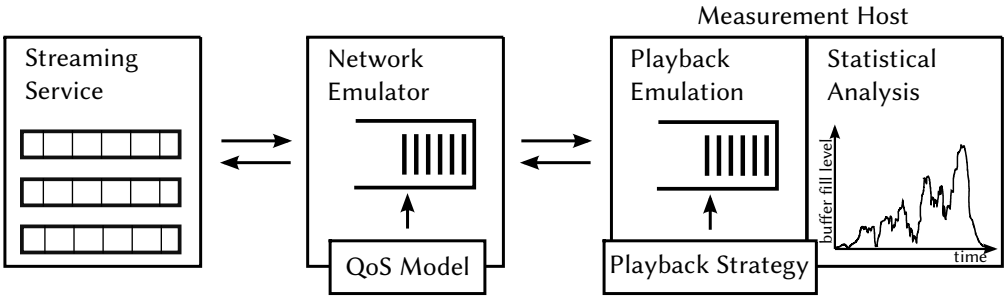


Figure 4.9: Overview of the measurement framework for adaptive streaming playback strategies.

Figure 4.9 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 allowing for a wider range of playback strategies, this approach is also inherently slower as it does not allow individual measurements to be 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 6.2.

4.4.3 *Technical Implementation*

To conduct actual measurements, the described two phase progressive streaming measurement framework has been implemented as a network testbed. The three individual components of the framework in Figure 4.8 are represented by three interconnected physical nodes running Linux.

The streaming server houses an Apache httpd Web server[11], hosting the files that are to be streamed. Alternatively, traffic from any viable Internet streaming service can also be directly routed through the network emulation node, making the local streaming server superfluous.

The network emulation node uses 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 will be set to a deterministic value for the following experiments. The loss follows a uniform distribution without any correlation in the transmission.

Curl[12] is used to both retrieve the streaming file and record the transmission process at the client node. If so desired, tcpdump[13] can also be facilitated to achieve a higher recording precision. The video file is then parsed for its frame timings and sizes using mplayer[14] with FFmpeg[15].

The traces are then put into the actual playback emulation, which can be run on any computer. It is implemented by custom Python-based code and statistically evaluated with Python[16] as well as R.

4.4.4 *Measurement Series and Evaluations with the Framework*

This testbed is now 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, specifically latency and loss. The evaluated strategies are the described YouTube and Firefox strategies as well as the null strategy and the predictive strategy with perfect knowledge and an optimally-sized initial pre-buffering phase that compensates for every degradation in the network conditions.

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 4.3.

11 https://httpd.apache.org/
12 http://curl.haxx.se/
13 http://www.tcpdump.org/
14 http://www.mplayerhq.hu/
15 https://www.ffmpeg.org/
16 The emulation code is publicly available at https://github.com/fmetzger/thesis-bufferemulation.

Table 4.3: Parameters of the video used in the streaming emulation measurement series.

Parameter	Value
Video duration	92.5 s
Size	9.61 MiB
Frame rate	$23.976\,\mathrm{s}^{-1}$
Average video bitrate	$871\,\mathrm{kbit\,s}^{-1}$
Codec	AVC

Two measurement series are performed with this video, both only differ in the network emulator settings. 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 Digital Subscriber Line (DSL) value of $16\,\mathrm{Mbit\,s}^{-1}$ in the downlink direction and $1\,\mathrm{Mbit\,s}^{-1}$ up.

It can be stated that all playback strategies will generally work similarly well under good network conditions as long as the TCP "goodput", i.e., the rate at which 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.

However, if the achievable throughput is close to the average video bitrate, the buffer can be quickly drained by short deviations from the average rates. High latency and loss are the typical limiting factors for TCP as many congestion control algorithms depend on the Round-Trip Time (RTT). If the RTT is high, the congestion window will increase less quickly. High latency can also trigger timeouts and retransmissions, which in turn decrease the congestion window again.

Packet loss can affect 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 stall 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. Further influences will be discussed in Chapter 5.

4.4.4.1 *Latency Measurement Series*

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. Each individual experiment was also replicated five times and corresponding error bars are provided in each figure.

Figure 4.10 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 a slight edge under high latency. Overall, the stalling duration quickly reaches a length comparable to the actual video duration and even surpasses that. Someone watching a stream under these conditions might find this not acceptable any longer.

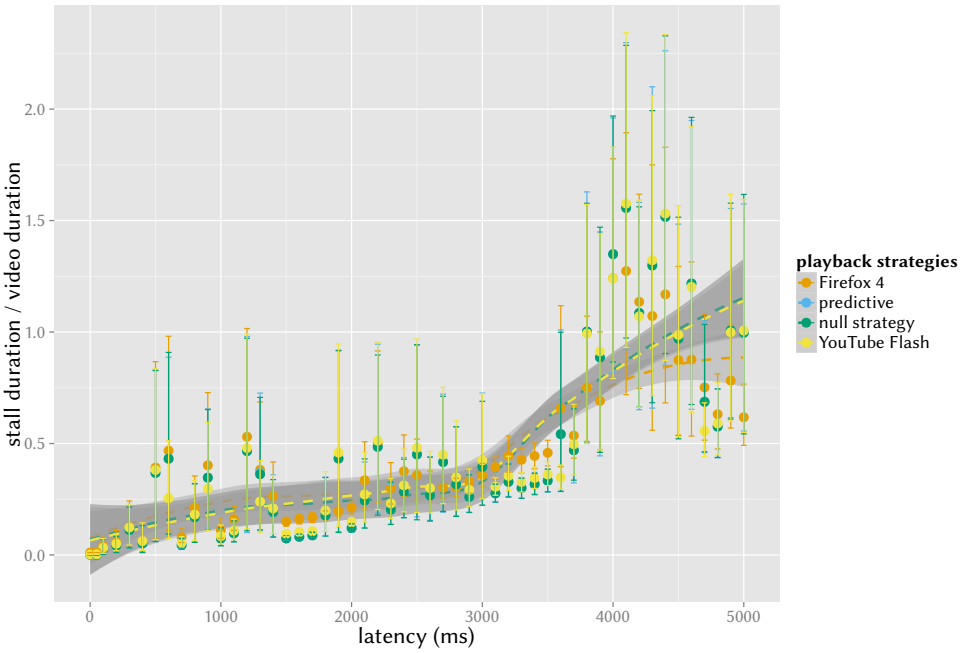


Figure 4.10: Stalling duration in relation to transmission latency with a polynomial least-squares fit.

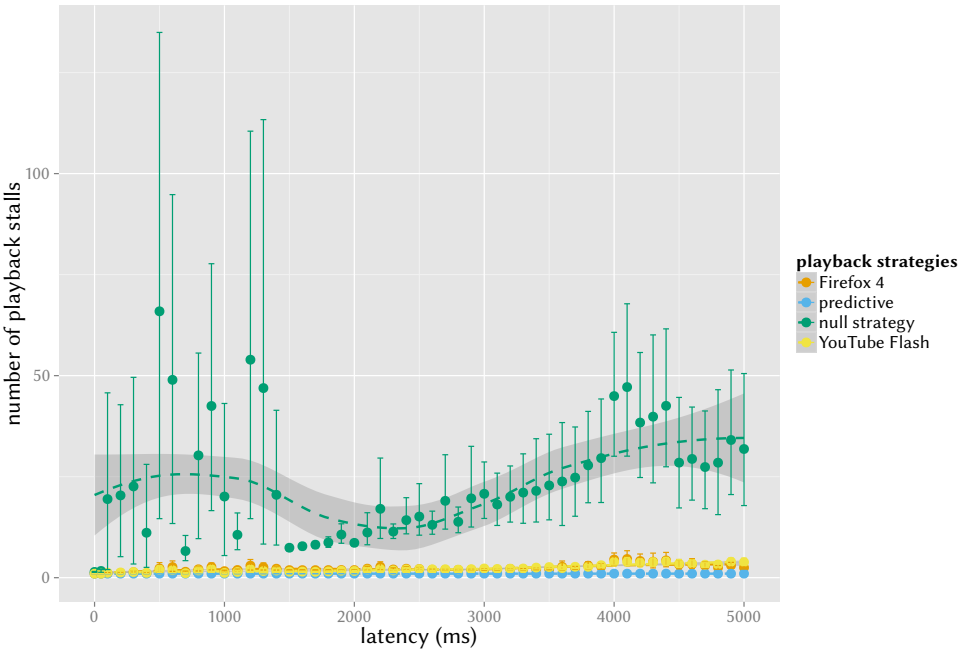


Figure 4.11: Number of stalls in relation to transmission latency with a polynomial least-squares fit.

Figure 4.11 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.

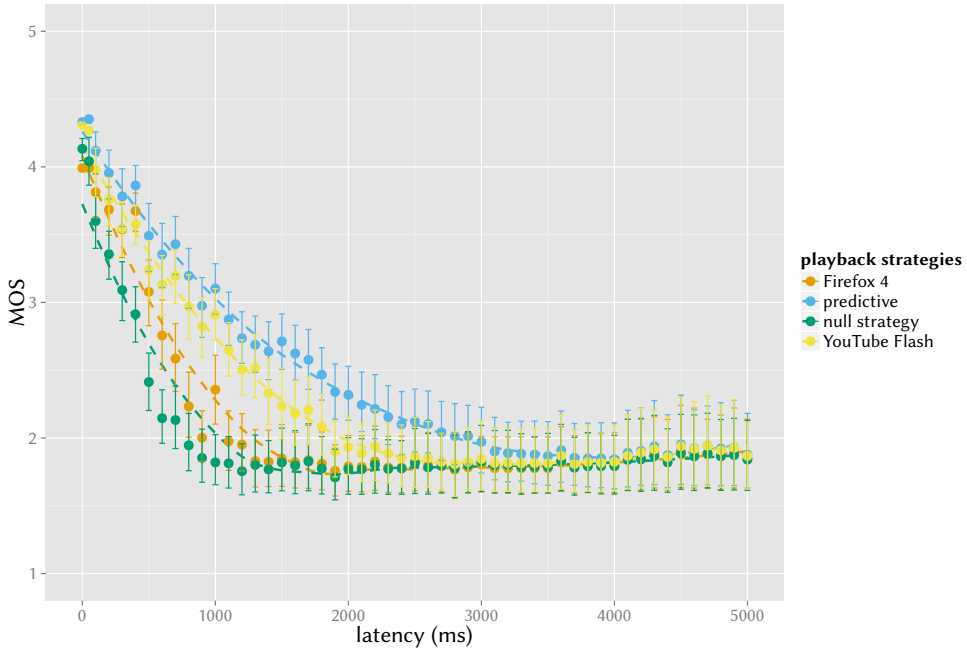


Figure 4.12: Calculated MOS for the latency measurement series.

Using the model from Equation 4.6 the QoE is calculated for this series and depicted in Figure 4.12. The MOS quickly drops below a value of 3, which is generally accepted as still being of fair quality, and stays around 2 for the most portion of the latency series. Starting at around 3000 ms of latency all four strategies achieve virtually the same poor quality, according to this model. But especially between 1000 ms and 2000 ms the difference in quality of these strategies is visible, with both the theoretical predictive and the YouTube strategy reaching a much higher MOS than the other two.

Overall, it can be said that in this specific latency scenario the real world playback strategies seem to honor the fact that more video interruptions lead to a worse experience than fewer but longer stalling events. Through mobility and handovers mobile devices fairly commonly experience short bursts of latency of several seconds. According to the measurement series, the resulting stalling behavior could still very well be bearable for streaming users if the latency does not reach too high values on average. For example, at 1000 ms latency a MOS of 3 could still be easily achievable with the right choice of playback strategy.

4.4.4.2 *Loss Measurement Series*

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 0.5 % 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 too many packets were lost back to back, and curl giving up after several retries. This unpredictability 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. Again, five experiments for each entry with corresponding error bars are conducted.

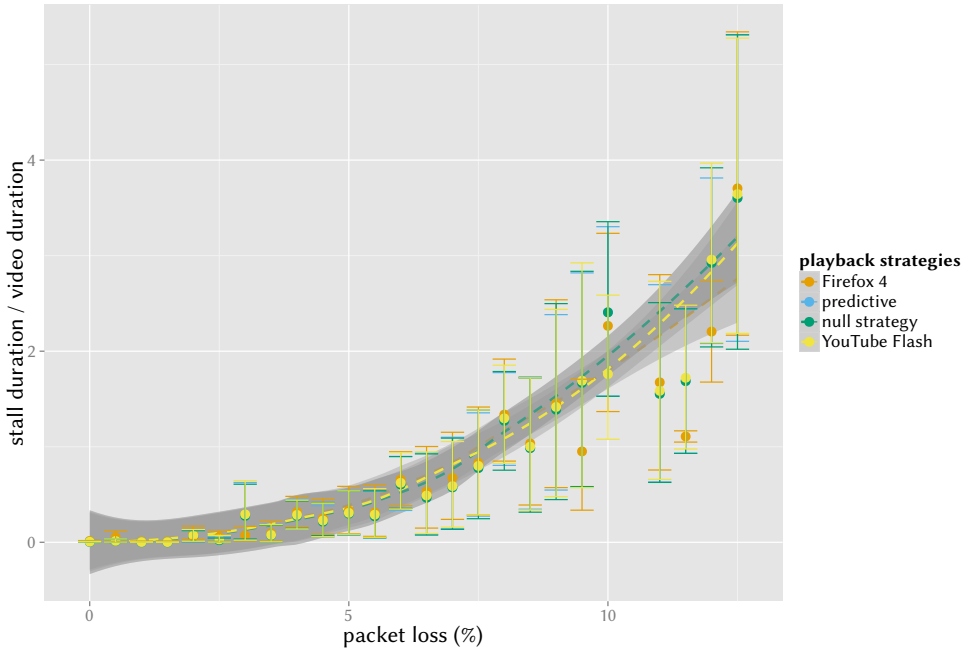


Figure 4.13: Stalling duration in relation to the packet loss with a polynomial least-squares fit.

Figure 4.13 shows the resulting relative stalling duration in the packet loss measurement series. Loss of up to about 2.5 % 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 12.5 % packet loss any streaming attempt is practically rendered unusable. Also, all four tested playback strategies handle high loss equally unsatisfactory, with the exception of some Firefox results producing a lower stalling duration.

This general behavior could be explained by the transport protocol's reliable transport feature, catching any occurring loss. However, the detection and retransmission of lost segments takes

time and leads to a bursty increase in latency. It also represents a possible reason of the increased stalling time.

At least it can be safely assumed that in actual production networks high values of packet loss are usually less likely to occur than high latency. The only major source of packet loss should be network congestion, which should only occur in moments of high network load. Therefore, it can be expected that playback strategies are better optimized for scenarios with high latency than loss.

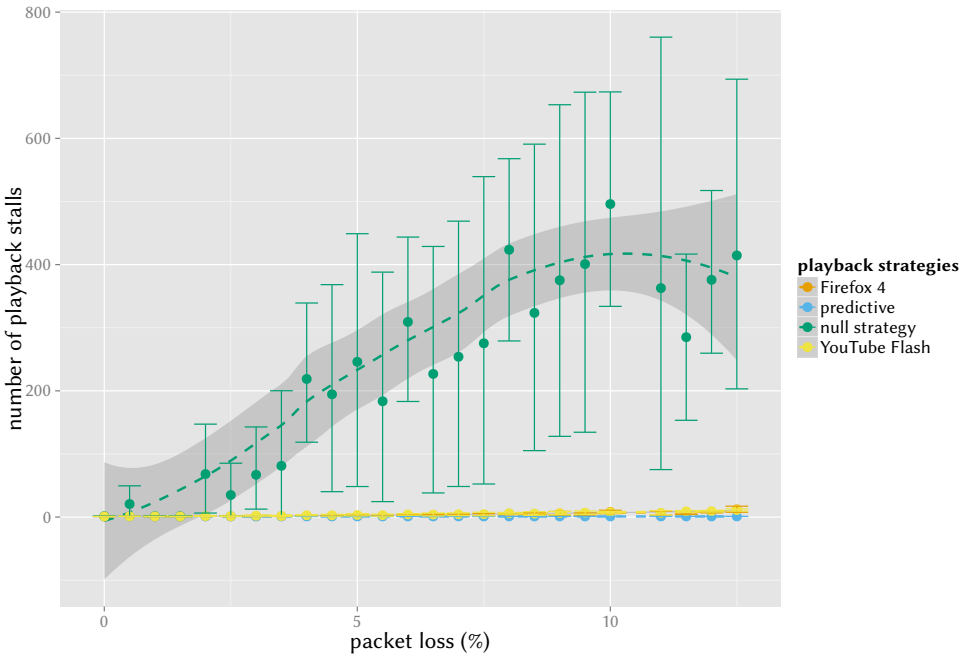


Figure 4.14: Number of playback stalls in relation to packet loss with polynomial least-squares fit.

Figure 4.14 clearly shows the extremity of the null strategy in terms of the number of experienced stalls compared to any other strategy. The same loss measurement series is used as basis here. The null strategy runs into two orders of magnitude more stalling phases than the three other strategies. The number of stalling events of the three other strategies remain relatively low. But the individual stalling events will be rather lengthy ones when keeping the total stalling duration in mind. Therefore, in the packet loss scenario the factor that will degrade QoE the most seems to be the duration of the stalling events but not their number, with the exception of the null strategy.

Looking at the QoE of the loss series, displayed in Figure 4.15, the difference in MOS between the playback strategies is much smaller than the one observed in the latency series. Additionally, the MOS degradation happens much more slowly and evenly dispersed across the loss values. A MOS of 3 is only undercut at around 5 % packet loss. But the fluctuations between the individual experiments is rather large in this series, often giving results 2 MOS units apart for the same

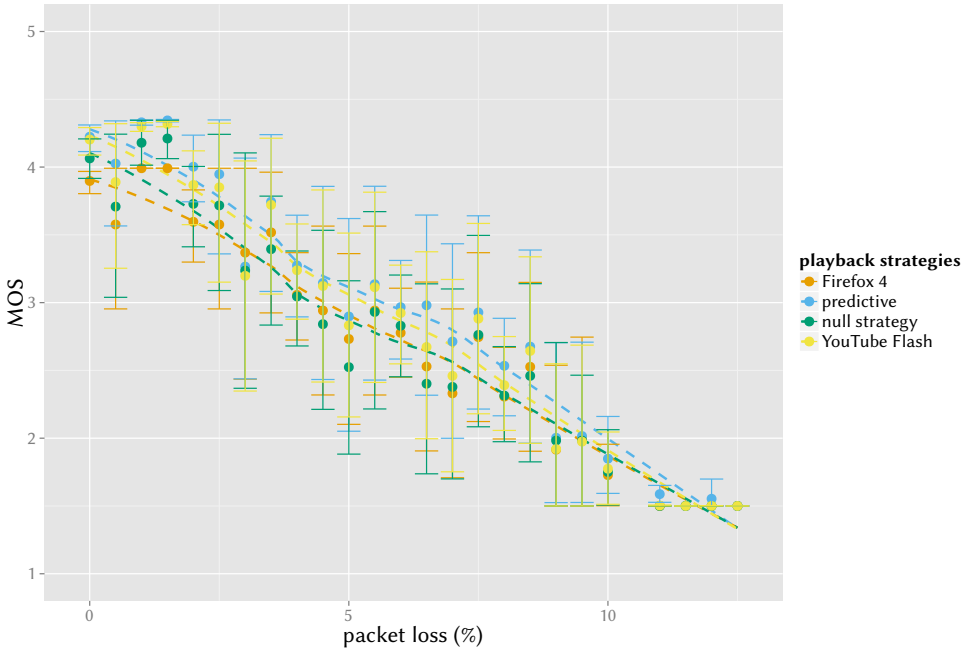


Figure 4.15: Calculated MOS for the loss measurement series.

packet loss percentage, making loss a very unpredictable factor as it often triggers secondary effects.

All in all, when planning a network for streaming applications, the maximum loss should be kept below the 2.5 % mark to achieve reasonable streaming quality. All the existing strategies already seem to work rather well for typically experienced network QoS scenarios. Only when extraordinary network conditions are present the strategies break down. But this is not different to many other network applications, which work best with pristine QoS.

4.5 Reliable 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 the creation of specific tools. Previous advances were generally not flexible or fast enough to deal with the influx of new system parameters. The presented measurement framework, based on a fundamental playback model, can be an answer to this issue. Implemented as an 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 was 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 devices, for the growing mobile application ecosystems, and for the very distinct behavior of streaming in mobile networks in contrast to classical wireline networks. The description of the mobile streaming environment and approaches for an investigation will be conducted in the next chapters.

5

RELIABLE VIDEO STREAMING IN MOBILE NETWORKS

The previous analyses of mobile networks were entirely based on having access to mobile core network measurements. Most research groups will in most likelihood not have this access and have to rely on active end-to-end device measurements or other approaches. Both the load characteristics of mobile networks as well as reliable streaming models were independently investigated up until now. But, as discussed in the beginning of this thesis and suggested in several mobile Internet traffic analyses and forecasts (see for example [Cis14]), video is already occupying a large portion from mobile traffic and might rise to an even higher ratio in the near future.

Now that the two independent foundations were laid out in the previous chapters, the next two chapters once again merge mobile networks and reliable streaming with two further lines of discussions.

First, the existing higher layer protocols of a mobile device's reliable video network stack and the influences of lower layer mobile layers on them are analyzed. Contrary to the popular scientific belief that the Internet is unable to change and renew itself (so called Internet "ossification", compare, e.g., [Fel10]), protocols in the TCP/IP stack change all the time. Changes are more likely to occur in the end-to-end stack behavior and not in the packet format, as the latter is more difficult to change. New application requirements and changing network architectures always trigger adaptations in the stack in-between. Therefore, new and upcoming protocols are presented with an additional focus on the rapid changes to the existing network stack that occurred in the past years.

This also ties in with the mobile network load influencing factors discussed in Section 2.3. Any behavior exhibited in the device's stack could also have positive or negative effects on the network's control plane and signaling procedures. The section concludes with a theoretic cross-layer information exchange model that has the potential to improve reliable streaming in scenarios with fluctuating mobile network connections, e.g., during mobility, by informing the streaming application about it and making it possible to react upon such events.

After that, in Chapter 6 several approaches to monitor and measure end-to-end mobile network traffic, and streaming video in particular, are demonstrated and executed. Having no access to in-network measurement probes limits the level of detail one can deduce from influence sources inside the network and one has to shift its attention to active measurements. But this also gives access to a wide range of other device-based information sources, some of which are crucial to mobile devices, especially when regarding mobility. These methods are also not limited to particular mobile network architectures and can thus be used to rapidly compare new architecture evolutions or specific core network implementations to each other. Finally, a full video streaming mobile network simulation model is also presented with some initial results.

5.1 Influences of the Existing Stack

This section briefly covers the existing Web protocol stack, which is also used for reliable streaming, in relation to mobile networks and streaming, new protocols and how they might influence the stack, and also discusses TCP more closely as an example. Beneficial interactions between the stack's layers are presented in a final part.

Superficially, not much has changed in the Web's protocol stack. There is still IP, TCP, and HTTP forming effectively the same stack since the development of HTTP/0.9 in 1991. And this seems hard to change. Especially the transport layer is fixed to UDP and TCP, everything else will probably get rejected, altered or even dropped by one of the numerous middleboxes, such as NATs, or forced "traffic optimization" transparent proxies, all of which are prevalent in mobile networks [Wan+11].

Each protocol, representing a layer in the stack, characteristically contributes to influencing data transmissions and varies in its degree of impact on the network as well as the intended goal of the transmission (e.g., streaming and watching a video).

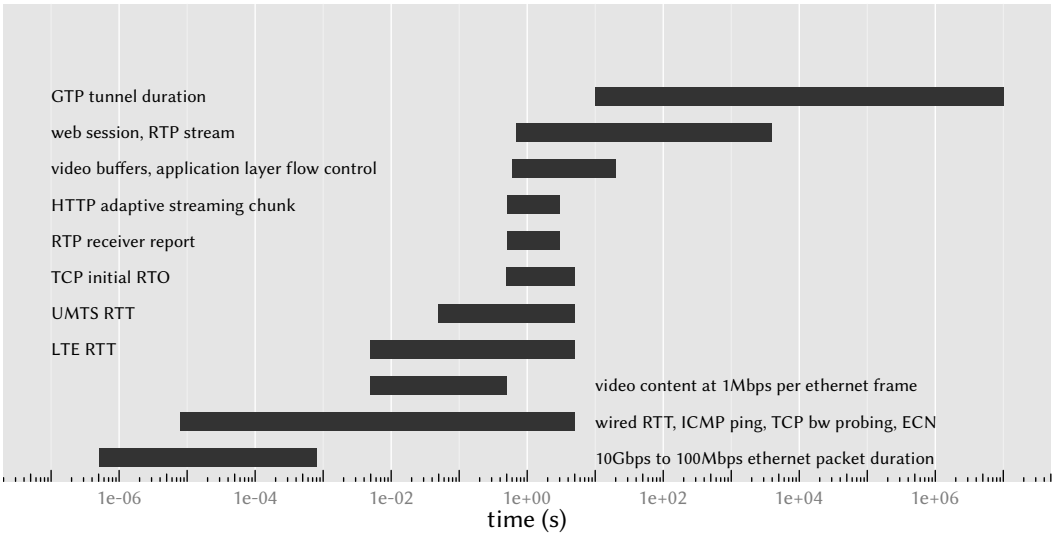


Figure 5.1: Approximate discernible time scales the networking stack protocols operate on in each layer.

Associated with each layer, and with the actual application running on top of the stack as well, are different timing constraints or time constants of control. Figure 5.1 overviews the approximate time scales on which activities take place, spanning a remarkable range of twelve orders of magnitude. Complexity is introduced by stacking multiple layers on top of each another, with many layers bringing along their own notion of control and feedback loops. Functionality might be duplicated across different layers, e.g., flow control in the application and on transport layer, leading to nested control loops, which might be coupled due to the timing constraints.

From the application layer end-to-end perspective the mobile network protocols seem to simply increase the depths of the protocol stack. This introduces effects intrinsic to wireless access and differentiate mobile networks from a fixed bit pipe.

Typical effects of wireless connectivity relating to physical phenomena like fading and interference can be observed. Interruptions in the radio link are a major source of packet loss and spiking delay. The overall delay in a mobile network is also strongly dependent on the mobile network technology in use and has considerably decreased over the last few 3GPP specification evolutions, even in the core network. This was investigated in [Lan+11]. But the RAB and GTP tunnel setup — as previously described in Section 2.1 — is even with the latest iteration of specifications sufficiently lengthy to be measurable and influences packet delay during connection initiation as was observed in [AF10].

To save radio and network resources devices usually have to quickly release their alloted radio resource slots causing additional delay for applications during connection reestablishment. Moreover, mobile networks offer many control options and parameters that can further influence any of the upper layers. Mobile devices have the tendency to be somewhat sensitive to some of these changes. The authors of [Wan+11] show that mobile devices are sensitive to the impact of protocol manipulations conducted by middleboxes.

Comparing all these factors to wired protocols such as 802.3 Ethernet it is clear that much more can happen in the mobile stack. Today Ethernet is a point-to-point protocol, transmission delay depends only on very few predictable factors, such as the distance. Wired access is typically not conducted solely through Ethernet, but with access technologies such as DSL using Point-to-Point Protocol over Ethernet (PPPoE) or cable with DOCSIS. While they also add more layers to wired access, they should be by far less influencing than the mobile stack. The latter has to additionally manage mobility and the shared wireless radio medium with protocols such as RLC and RRC.

A further detail can be found in the interaction between mobile network frame sizes and the IP Maximum Transmission Unit (MTU). Usually IP adapts its packet size to the shortest frame size of all the link layer links in the path using Path MTU (PMTU) discovery as described in [RFC1191]. Unfortunately, this can not work in 3G mobile networks, user packets are transparently fragmented by the radio protocols to fit into the transmission slots alloted to the device. For example, a GPRS transmission slot has a length of only 576.9 µs carrying just 114 bit. And in UMTS 40 B of payload are typically carried in each RLC frame (with optional header compression in the PDCP layer).

This fragmentation is another source of an undesired interaction between 3G's link layer and everything above and including the network layer, potentially fragmenting packets over a long period of time. From an application point of view this will cause additional packet delay.

In a neutral network, all packets are treated the same and should be subject to the same RTT influencing sources. However, delays in the delivery of certain data to the upper layers can occur due to certain protocol features, such as TCP retransmissions. After being detected through timers or duplicate acknowledgments any lost packet is automatically retransmitted by the sender. Therefore, loss from the lower layers is hidden from the application layer at the

cost of delay variations and probably increased overall delay. The increase in delay is especially noteworthy in a 3G cellular network with its notion of link layer Automatic Repeat Request (ARQ) and mobility. With the latter concept, transmissions originally running through one radio tower have to be moved to a new tower after a handover, even if the handover period was lengthy and the data might already be outdated. These two combined can be the source of very high delay, reaching seconds or even hundreds of seconds, when moving quickly between coverage areas (e.g., in a car or train). This also causes bad interactions with TCP, as these long-delayed segments are thought to be lost and subsequently retransmitted, resulting in an even further increase of delay and delay jitter.

Another undesirable influence factor is caused by each packet buffer in the transmission's path. The effect occurs if the size of the buffer is not chosen carefully and the following link is a bottleneck link. Additionally, memory is very cheap, so hardware vendors tend to plug in as much as they can. Packets may be kept unnecessarily long in these buffers, heavily increasing packet delay and diminishing TCP's ability to adapt itself to its fair share with the congestion control mechanisms [Jac88; Sch11]. These mechanisms rely on feedback from the network by inducing packet loss which does not happen in a timely manner in case of large buffers. This phenomenon has been dubbed *Bufferbloat*, and can induce latency as high as several seconds [GN11; GK11]. This has to be considered in addition to the latency in mobile networks which is already high and unreliable to begin with.

Buffers are necessary to compensate for short-term variations in the traffic load and avoid loss in these situations. A popular countermeasure are Active Queue Management (AQM) methods that control the size of the buffer and selectively drop packets if necessary. Although, most of these are never widely used because of configuration issues and their inability to automatically adapt changing network conditions. One particular algorithm stands out that is becoming increasingly popular: CoDel [NJ12; NJ14].

CoDel measures the time it takes for a packet to pass through a buffer and begins dropping packets once a certain maximum packet service time has been exceeded for a predefined timespan. This and other similar mechanisms that control queue delay rather than the queue size have begun to appear and be used in several places, e.g., in the Linux Kernel.

Up to the point of the transport layer the layers are pretty well defined, with only a few select protocols to choose from, with mostly known influence factors on the mobile network. The application layer has a different notion, though. Adhering to the end-to-end design principles most innovation (and therefore change) will take place at the utmost ends, i.e., the application layer. Even when only considering video streaming applications the protocol variations are vast as the overview and classification in Section 4.1 demonstrated.

In streaming over mobile networks the video buffer size has to be in the range of seconds to compensate for most eventualities as seen in Figure 5.1. Also every streaming protocol has slightly different time scales and feedback loops.

As is the case in any best-effort network no bandwidth, latency, or loss guarantees can be given. The available bandwidth might get unexpectedly smaller resulting in a quickly draining video buffer if it was chosen too small. Then again, given sufficient bandwidth stability or

at least predictability, buffer sizes could also be kept at a bare minimum, thereby enabling or improving real-time interactivity. In addition to a fitting choice of the buffering and playback strategies the choice of segment length and quality levels in adaptive streaming also plays an influential role in mobile networks. Most of these factors were discussed in Chapter 4.

In layered network models individual functions are strictly separated and stacked on top of each other with well-defined APIs connecting them. To achieve complete separation without cross-influences, the feedback and control loops of each layer also would need to operate on completely disjoint time scales. Looking again at Figure 5.1, it can be seen that this notion is only partly supported and the loops of adjacent layers typically overlap somewhat. Protocols and implementations need to be aware of this circumstance and able to handle cross-layer influences.

5.2 Recent and Upcoming Protocols

Protocol development has not stopped in recent years. On the contrary, especially in the application and transport layers some interesting changes are occurring. The following paragraphs highlight some of the changes and their potential influence on or improvements to mobile streaming.

Beginning at the transport layer some alternative approaches to TCP are available, as especially the retransmission reliability has proven to be problematic in mobile networks as discussed. Typically, TCP is used regardless for two reasons: First, having congestion control is an absolute necessity to achieve an equal fair share bandwidth and avoid congestive collapse. If no congestion control is present on the transport layer it would need to be implemented elsewhere. This generates additional implementation work and simultaneously reduces portability and compatibility to other congestion control variants — the term *TCP friendliness* has been used for this in the literature.

Second, due to many middleboxes simply dropping packets using any unrecognized protocol — exempting UDP and TCP — the transport layer protocol often can not be changed at will. New transport protocol approaches would need to layer themselves atop UDP incurring additional overhead. These following three items are examples of such transport protocols, each created with different goals in mind:

- DCCP [RFC4340] extends UDP with concepts for data flow and TCP-compatible congestion control but omits a strict packet retransmission feature.
- The Low Extra Delay Background Transport (LEDBAT) [RFC6817] congestion control algorithm which is implemented, amongst others, in Micro Transport Protocol (μTP) [Nor]. Traffic using LEDBAT is generally of high volume, high elasticity, and low priority, such as for example file sharing traffic. Data handled by this congestion control approach gets displaced by any other data if the algorithm detects an increase in the transmission buffer delay. These aspects makes it rather unsuitable for video streaming which generally does not have elastic properties.

- Quick UDP Internet Connections (QUIC)[1] is a further experimental approach aimed at reducing sources of latency spikes which would occur in TCP, such as retransmissions or the initial handshake. Some reliability is achieved through Forward Error Correction (FEC) schemes if desired. Furthermore, a novel congestion control mechanism based on packet pacing, not through a traditional congestion window, and encryption by default are implemented. However, it is not intended as a production protocol but is rather a testbed for future mechanisms and modifications to existing protocols.

Especially DCCP and QUIC show some interesting prospects for mobile streaming through their omission of retransmissions, which is the source of most playback stalling in high latency and loss scenarios. This would return streaming players a degree of freedom which they had while using the unreliable RTP. A playback strategy could now once again decide what to do in case of packet loss. The choice would be either to stall and wait for subsequent video data or drop some frames and continue the playback more rapidly.

In the wake of Edward Snowden's surveillance and man-in-the-middle reveals, the IETF announced its position on these issues in [RFC6973] suggesting that all future standardized protocols should consider surveillance issues and minimize or prevent any attempt on it. Ultimately, this might lead to a end-to-end *crypto-by-default* Internet, with every protocol providing at least opportunistic encryption. The first signs are already visible in today's traffic mix, with many Websites only available through HTTPS — using Transport Layer Security (TLS) [RFC5246] — any more. Similarly, most e-mail providers have moved to IMAPS and SMTPS, even using new authentication and key exchange techniques like DNS-Based Authentication of Named Entities (DANE) [RFC6698] which in turn relies on DNS Security Extensions (DNSSEC) [RFC4033] to authenticate the validity of DNS entries. For example, the current draft version of HTTP/2 — which will be described in more detail shortly — mandates the presence, albeit not the usage, of TLS capabilities in both client and server. And even the connectionless UDP can be secured using Datagram TLS (DTLS) [RFC6347].

Usually, an additional intermediate layer is added through the encryption process, adding more timing interdependencies to the other layers. However, end-to-end encryption also diminishes the influence of most middleboxes as they are not able to alter the contents of the — now encrypted — segment any more. More and more Web streaming services are also available through HTTP Secure (HTTPS), which causes a large shift of the traffic mix to encrypted protocols. Even for RTP encrypted variants, in the form of SRTP [RFC3711] and ZRTP [RFC6189] are available and are being commonly used for communication applications.

Moving on to the application layer, some interesting developments have occurred in the Web's stack and in its use of HTTP which almost all reliable streaming methods facilitate.

The original HTTP/1.0 specification is from the year 1996 with only one update to HTTP/1.1 in 1999 thereafter. Since then, no changes have been made to the core protocol. However, the number of use cases facilitating HTTP has increased significantly, bringing with them a lot of long-standing issues that were not thought of initially.

1 http://lwn.net/Articles/558826/ and http://www.ietf.org/proceedings/88/slides/slides-88-tsvarea-10.pdf

The typical number of individual HTTP requests for one Web page has steadily increased, reaching today about 100 objects.[2] HTTP deals with this through persistent connections and pipelining to avoid unnecessary connection establishment and request delays. Pipelining is however disabled in almost every client implementation as it suffers from head-of-line blocking. To circumvent this problem and increase retrieval speed, browsers have begun to open many parallel TCP connections per site, invoking significant overhead and general fairness issues due to the large number of TCP connections displacing other traffic. Segment-based reliable streaming applications faces similar issues, especially when more than one segment is to be retrieved simultaneously. In the worst case, this can add additional latency through the need to establish further connections.

Moreover, the Web's structure has changed in such a way that it is now common to push data to the clients to increase a Web page's interactivity. However, HTTP is a simple client-initiated request-response protocol and pushing can only be emulated using techniques such as *long polling* which imposes several restrictions and state overhead. Both these issues also affect reliable streaming with HTTP to a certain degree. Streaming can benefit from a reduction of request latency as well as servers pushing adequate video segments at the correct time without the need for further requests.

To combat the situation and increase the protocol's interactivity along with it, several approaches are being developed. The first, WebSocket [RFC6455], is a transport protocol on top of TCP (or TLS) that can be established through a regular HTTP connection. It enables full-duplex communication between the two communication nodes. An additional W3C Application Programming Interface (API) [Hic13] enables simple usage in browsers with higher interactivity than plain HTTP. It is also used to stream content to a browser.

The WebRTC protocol suite[3] [W3C13] even goes a step further and provides a complete stack for multimedia communication. It is intended to be used — through the provided API — for direct browser-to-browser communication initiated by a central server-side Web page.

Both these techniques aim to enhance or sidestep HTTP. But it is also desirable to improve the protocol itself for the aforementioned reasons. Therefore, in 2010 Google implemented a new experimental application protocol called SPDY as an alternative to HTTP in its Chrome browser accompanied by a draft specification [BP13; BP12]. Every major browser and most Web servers now support SPDY, many Cloud and CDN providers have it enabled.

SPDY uses the same port as HTTPS (443) and mandates the use of TLS as it also serves to negotiate the choice between HTTP and SPDY. Compared to HTTP pipelining, SPDY provides full-duplex time-division multiplexing. After an initial request of a Web page, the server can anticipate any follow-up requests, push objects on additional streams with assigned priorities and therefore avoid additional request round trips.

All of the SPDY changes are aimed at reducing the number of established TCP connections for a Web site retrieval, ideally down to one, and reduce the overall latency of the object

2 http://www.websiteoptimization.com/speed/tweak/average-web-page/ and http://httparchive.org/trends.php and https://developers.google.com/speed/articles/web-metrics

3 http://www.webrtc.org/

transmissions. Evaluations in [BP12] suggest an average Web page load time reduction of 29 %. For networks with high latency, e.g., mobile networks, the gain could be even higher as several round trips are avoided.

Some of the benefits are also applicable to reliable streaming, in that servers are now able to push video segments to the client. Conjoined with Scalable Video Coding (SVC), SPDY could multiplex the different video layers in one connection, with the highest priority assigned to the base layer.

The SPDY draft was later picked up by the IETF Httpbis working group as an initial basis basis for the HTTP/2 specification [BPT14] currently being finalized as of mid 2014. It follows the SPDY specs rather closely, with the only major difference being the use of TLS. Its usage is not mandated anymore, merely its presence and, if used, the application of strong and ephemeral cipher suites.

In fact, Netflix, one of the largest commercial video streaming providers, already utilizes the features of the latest HTTP/2 draft in its service. For example, segments are pushed by the server using HTTP/2 multiplexing and are displayed as HTML5 video. The usage of these features for video streaming might merit a separate evaluation.

5.3 Additional TCP Changes

In addition to the previously described new transport protocols under development, aiming to replace and surpass TCP, TCP itself is not as static as it may seem. While the basic wire frame and data interchange format has already been defined in 1981 in [RFC793], much of the specific behavior is free to be experimented with by further specifications and implementations. The most prominent example is the congestion control mechanism where every operating system takes hugely different approaches.

Every one of these changes can have unforeseen repercussions or benefits for the application layer. The following paragraphs investigate some of the changes in the last few years and their implications. Instead of implicitly exploring specific mechanism choices of closed source OSs and tracking their changes over the years, which would be extremely difficult, the TCP implementation in the Linux kernel is taken as an example. The changes can be easily tracked by crawling through the patch sets and their commit logs of every kernel version. Sites like `http://kernelnewbies.org/` and `http://lwn.net/` simplify this process even more. Table 5.1 depicts such an attempt to track the major changes affecting TCP in the kernel.

Most of the changes plainly aim at reducing the number of required round trip times and increase the overall throughput while still maintaining fairness. This includes both small and larger changes, some of the more recent are:

- The TCP congestion window defines the maximum amount of data that can be transmitted without having received any acknowledgments for the data. It is the central knob for all congestion control mechanisms.

Table 5.1: Assorted list of some select network stack changes in the Linux kernel that alter TCP's transmission behavior.

Change	Related Work	Kernel	Date
BIC as default congestion avoidance algorithm (from Reno)		2.6.8	August 2004
CUBIC as default congestion avoidance algorithm	[HRX08]	2.6.19	November 2006
New TCP Slow Start: HyStart	[HR11]	2.6.29	March 2009
Multipath TCP	[RFC6824]	external	2011
TCP User Timeout	[RFC5482]	2.6.37	January 2011
Initial Receive Window 10 MSS	[RFC6982]	2.6.38	March 2011
Initial Congestion Window 10 MSS	[RFC6982]	2.6.39	May 2011
1 s initial RTO (from 3 s)	[RFC6298]	3.1	October 2011
Changes to sstresh and CWND bevahior	[RFC5681]	3.1	October 2011
TCP Proportional Rate Reduction	[RFC6937]	3.2	January 2012
Byte queue limits and TCP buffer limits		3.3	March 2012
CoDel AQM	[NJ14]	3.5	July 2012
TCP Early Retransmit	[RFC5827]	3.5	July 2012
TCP small queues		3.6	September 2012
TCP Fast Open (client side)	[Che+14]	3.6	September 2012
TCP Fast Open (server side)		3.7	December 2012
TCP tail loss probe		3.10	June 2013
TCP Forward RTO-Recovery	[RFC5682]	3.10	June 2013
Low latency network polling		3.11	September 2013
Improved RTO calculation and handling of reordering		3.12	November 2013
TCP Fast Open enabled by default		3.13	January 2014
TCP auto corking		3.14	March 2014
PIE AQM		3.14	March 2014
TCP Fast Open over IPv6		3.16	2014
LISP	[RFC6830]	3.16	2014

The initial window size was increased multiple times in the past — from initially one segment, to three, and now to ten segments [RFC6982] — with the intent to accommodate for the increase in transmission bandwidth and higher expected traffic volume. With a ten segment window small HTTP objects can be transferred without any ACK round trips reducing the average Web page load time. The initial window is also important for determining the achievable throughput in segmented streaming as the congestion window is often reset in the idle phase between segments.

- Another TCP round trip can be saved by using the initial handshake itself for data transmission. TCP Fast Open [Che+14] implements this and is again designed to reduce Web page load times.

- While the retransmission timeout is calculated through RTT measurements, it is initially set to 3 s. This could add a high amount of delay if packets are lost during the handshake at the beginning of a connection. Implementations following [RFC6298] set the default value to 1 s which may benefit especially mobile networks.

- The classic congestion avoidance algorithms do not work very well with networks with a large Bandwidth-Delay Product (BDP) as the congestion window scaling entirely depends on the RTT. Newer approaches reduce this dependence while also departing from the traditional linear growth congestion avoidance phase. Some of the congestion avoidance in use in today's OSs are:

  - Reno [RFC5681]
  - New Reno [RFC6582]
  - Vegas [BOP94]
  - Compound [SZS06] (as an option in recent Windows versions)
  - CUBIC [HRX08] (default in Linux)

All of these changes should serve to demonstrate the flux TCP is in. It is constantly being adapted to current network and application needs. However, no adaptation or assumption as to specific applications are made on the transport layer. These kind of modifications can only be made on an end-to-end basis, i.e., at the application layer (cf. also the arguments in [SRC84]). All lower layers must be application-neutral as they could harm other application layer protocols.

All these changes contrast the claims that the Internet is ossified and cannot accommodate any new applications any more (confer for example in [TT05]). Rather, IP and TCP serve as the common carrier of every type of data at the waist of the Internet's protocol hourglass model.

It also demonstrates the difficulties and large optimization space reliable streaming has to deal with. Finding the correct combination of protocols for every layer, parameters for these protocols, and playback strategies for every user scenario is a very complex task and needs proper tools to handle this.

5.4 Cross-Layer Information Exchange

The Internet has its historic roots deep in wired networks with a slim and well-defined network stack represented by the ISO/OSI or TCP/IP model. These layers are isolated against each other. Only predefined information exchange points, or Service Access Point (SAP), at the layer borders allow for vertical communication. A typical wired TCP/IP Internet environment rests atop of either an Ethernet or other access technologies, e.g., DSL, DOCSIS or PON, at the physical and link layers.

Application layer protocols often implicitly rely on the presence and characteristics of specific lower layer protocols, although through the layer isolation no application can precisely know or even control the current state of the lower layers. Nonetheless, they usually make assumptions on the composition and behavior of the lower layers and plan their work accordingly.

But the access technology diversity has strongly increased through the advent of wireless technologies, and fixed access behavioral patterns, which were examined in the past, may not be applicable any more today. The protocols used for the radio transmissions behave very differently when compared to plain Ethernet and higher layers may make false assumptions. Examples for this were given in the discussion of the stack's influences in Section 5.1.

It would be very desirable for transport and application layer mechanisms to be able to better understand these layers and cope with these effects. The term *cross-layer interactions* or *cross layer information exchange* subsumes these approaches. Specific information from one layer is made available to other interested neighboring or more distant layers. Using cross-layer techniques many of the previously introduced negative layer influences can be diminished or neutralized altogether.

The next sections describe related mechanisms in the literature, classifications and then proceed to describe a new cross-layer approach that facilitates cross-layer information to the benefit of mobile streaming. The cross-layer work presented here is meant as an initial proposal to be integrated into future streaming players and their playback and transmission strategies. Therefore, no complete implementation or evaluation is given.

5.4.1 *Related Cross-Layer Approaches and Classifications*

The idea of exchanging information between layers is not a particularly new one. Some specific ideas have been implemented a long time ago. The authors of [RI04a] list a number of scenarios in which cross-layer information could be used and also talk about the type of information to be shared between layers. One of the oldest and most well known cross-layer approaches is probably Explicit Congestion Notification (ECN) [RFC3168]. Here, the IP layer of intermediate hops can signal the end nodes' TCP layer that congestion is occurring and TCP does not need to wait for implicit congestion signals, e.g., duplicate acknowledgments or timeouts. However, ECN is disabled in almost every implementation by default as it lead to numerous problems and triggered bugs[4]. This is a risk that many cross-layer attempts may face.

4 http://lkml.iu.edu//hypermail/linux/kernel/0009.1/0329.html

A significant amount of publications is dealing with cross-layer information in wireless and mobile protocol stacks. A number of architectures have been proposed, e.g., [RI04b], [RI06], [WA03b], [WA03a], and [KI07], but no actual solution seems to have been implemented in any of these.

While cross-layer typically implies a solely **vertical** — meaning between network layers — exchange flow there can also be **horizontal** — between network nodes — components present. Dynamic Link Exchange Protocol (DLEP) [Rat+13; BR12] is such an example of a **diagonal** flow, providing information of a lower layer of one entity to a higher layer at another node. For example, DLEP can forward information available only to the (external) wireless modem or other interfaces to the routing entity of a node upon which it can act. Link characteristics such as bandwidth, latency, connection status, or information regarding neighbors can be requested.

Horizontal information flow in cross-layer approaches is more than often an indication of **centralization** (also called **network-assisted** or **managed**) as apposed to purely vertical **distributed** approaches. Concerning this network-assisted cross-layer exchange there are a number of approaches that aim to integrate layer cooperation into the design of new mobile network infrastructures, including [Zar+10] and [Pia+11]. Generally, information is retrieved from the clients and collected at a central manager to be used in any policy decision like mobility and radio resource management.

Going back to purely distributed approaches, in [HHM10] the concept of mobility awareness is discussed. The goal is to predict device motion and mobility based on available information and adapt the individual network layers to react accordingly. One of the easiest mechanisms to implement the usage of cross-layer information for is the selection of the active network interface. Current mobile devices have a wide range of network interfaces available, all with specific characteristics. The management architecture proposed in [BLH09] switches the currently used interface based on pre-configured profiles. Information from multiple layers is used to support the decision-making process.

To optimize unreliable video streaming in a WiFi network, the authors of [Har+06] create a control loop between the video encoder and the 802.11 MAC layer to conduct WiFi rate control fitted to the output of the encoder. This is an example of a tight cross-layer control loop for one particular application. The link layer rate control will very likely have adverse affects on other applications using this node.

The notion of cross-layer can also be applied to non-traditional network stacks. For example, the authors of [Mel+08] present a cross-layer model for satellite communication stacks. They additionally distinguish between two general flavors of cross-layer architectures, one with **direct** and the other with **indirect** communication. Direct exchange implements new vertical interfaces in the layers and exchanges information directly between them. The indirect alternative uses an external information broker that handles all communication in parallel to the existing layers. Similar cross-layer information can be offered to peer-to-peer overlay networks, as was for example researched in the SmoothIT project[5], where routing and topology

5 http://www.smoothit.org

information was collected and made available to interested peers [Oec+09] in order to keep traffic local and avoid using ISP interdomain links.

5.4.2 *Cross-Layer Model and Implications*

With these past approaches and classifications at hand, a cross-layer model suitable for video streaming in mobile networks can now be defined and the information to be exchanged specified.

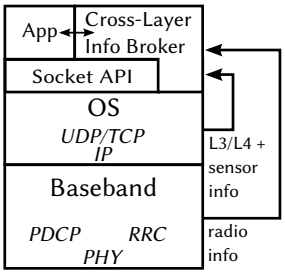


Figure 5.2: Model and architecture of the proposed cross-layer information exchange.

Figure 5.2 illustrates the concept with the example of a mobile device. In a fully isolated model, no information would be passed from the baseband to the OS and the applications. The cross-layer model permits certain information to pass from one layer to another. Here, a software broker is responsible to collect information from several sources and layers and make it available to any interested application in a concise manner.

This is an indirect cross-layer approach bypassing the intermediate layers completely and leaving them mostly unmodified. Also, information is always collected and used on the same device, making it purely vertical and distributed. Sharing this kind of information between different hosts would be accompanied by certain privacy and security implications. Most of the data is very sensitive and can also be used with malicious intent if it were leaked by the cross-layer broker.

This architecture is intended to pass data from the physical and link layer directly up to the application. Especially information specific to 3G mobile networks could be potentially interesting and used in benefit for applications and the user experience.This could include:

- Information on the occurrence of a horizontal handover between cells.
- Information on neighboring cells and predictions when a handover is most probable to occur.
- Information on the prediction of the occurrence of a vertical handover and thereby changes in the active network stack, e.g., to the WiFi layers.
- Information on the current signal strength, bit error rate, and throughput.

- Detailed mobility information, including current location and travel speed in relation to base station positioning and availability.

Today, most of this information is only available inside the link layer or even just known to the mobile network's control plane. The impact of a lack of this knowledge can be significant for applications. For example, traffic scheduled during a handover period can be subject to especially high latency and loss due to the lengthy control plane interactions and traffic rerouting processes in a mobile network. If a cross-layer exchange would be provided and the application is made aware when an handover is supposed to occur, traffic could be scheduled around the event. In addition to that, avoiding retransmissions and buffering during handover events would also help the mobile network save radio resources and buffer space at the mobility anchors.

Generally speaking, the goal of the cross-layer approach would be to find meaningful reactions for every type of state the lower layers reported through the broker. The pool of recipients is also not necessarily limited solely to the application layer. Especially the transport layer could be interested into explicit connectivity information and be modified to react accordingly. In addition to information flow, a path for control flow could also be envisioned. Herein, applications could directly influence the decision making and policies of the lower layers and adapt them to their personal needs.

All in all, both the cross-layer data needs as well as the recipient's reactions have to be well defined and thoroughly tested to avoid any conflicts and layer separation issues. The impact of a simple unidirectional information flow on the layering mechanisms is suggested to be rather low. Only explicit and specific information is revealed keeping most of the isolation intact. However, a bidirectional control flow could soften up the isolation and have adverse side-effects through conflicting interests of participating applications.

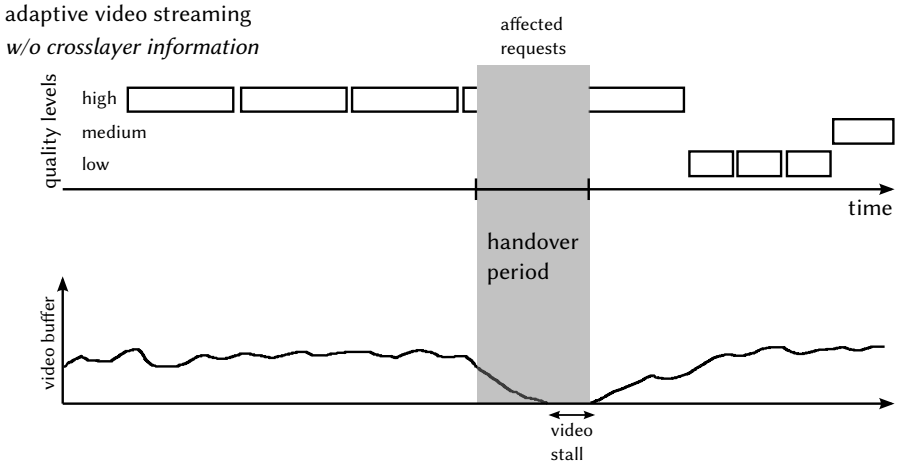
Either way, when implementing any kind of cross-layer exchange, one always has to keep a close look on the resulting consequences. One side effect can be the creation of an unintentional feedback loop between the control mechanisms of protocols of different layers. Moreover, breaches in the isolation could leak network state that could be exploited by malicious parties in any number of unforeseeable ways. Therefore, handling these plays an important role in cross-layer research. In [KK05] the authors present and discuss some of these issues.

5.4.3 *Utilizing Cross-Layer Information for Adaptive Reliable Streaming in Mobile Networks*

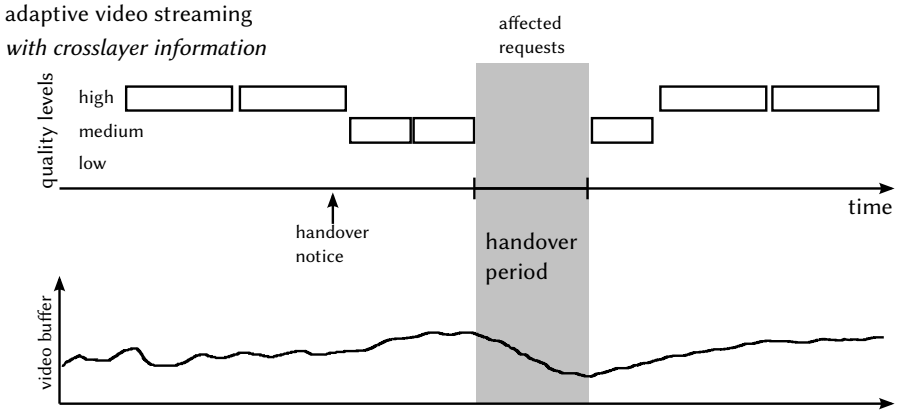
Looking at the model it can be an obvious fit for adaptive reliable streaming. As introduced in Section 4.1, adaptive streaming usually facilitates a segment-based pull approach on top of HTTP. A local video buffer is maintained and attempted to be kept between predefined thresholds. The goal is to never run out of buffered video while still providing the best possible quality, which could be difficult to achieve in a highly variable mobile network.

Cross-layer information can be fed into the adaptivity model of the streaming player to better decide the exact schedule and quality level of segment transmissions. Adaptive reliable streaming can be especially suited to receive cross-layer data for several reasons: First, all

requests are client-initiated, so available information can be immediately taken into account and does not need to be transfered elsewhere. Second, adaptive streaming consists of small and independent video segments, which allows to quickly react on upcoming events.



(a) Stalling occurs without handover hinting.



(b) Stalling can be prevented by hinting and proactively filling the playback buffer.

Figure 5.3: Adaptive video streaming scenario with and without handover prediction and cross-layer hinting.

One of the easiest events to improve upon is the timely knowledge (or prediction) of upcoming handover procedures, horizontal as well as vertical handovers. Consider the scenario given in Figure 5.3a. The streaming player consistently retrieves video segments at the highest quality, maintaining a moderate buffer size but is oblivious to the upcoming handover. This handover interrupts any transmission for a certain time, during which the buffer is fully drained and the video playback begins to stall. Only after this phase ends, the remaining portion of the segment can be received. But the buffered amount is now still below the safety margin and the player is forced to request several segments in the lowest quality to quickly refill the buffer. Only after that, the player's state normalizes and normal quality playback is restored. In summary,

one unexpected service interruption causes one complete stall and a long period of reduced quality in this exemplary scenario. This effect can also be easily observed in the mobile reliable streaming simulation scenario described in Section 6.2.1.3.

Figure 5.3b shows the same scenario but with cross-layer hinting present. As soon as the notice of a predicted upcoming handover is received, the streaming player can react and switch to segments with medium quality to increase the buffer size before the event. After the handover has completed one more medium segment is transmitted to get the buffer level back to normal before returning to full quality. Both the video stalling period and the drop to the lowest quality could be avoided here. The general goal in the streaming process is to both stop the buffer from ever completely emptying and also maintaining the highest possible video quality. To achieve this, early knowledge of future network conditions is highly desirable for the streaming controller to correctly adjust the segment retrieval rate and quality.

The end-to-end concept can be applied to cross-layer architectures as well. While cross-layer information can still be made available to intermediate layers, the achievable effects might either not be as large as in the highest layer or more complicated to attain. For example, it requires much more effort to correctly adjust TCP retransmissions and congestion control to accommodate the cross-layer broker without side-effects to other applications. Nonetheless, benefits can still be attained at the transport layer to some degree if the adjustments are being kept application-neutral. This can be especially helpful for applications that have not yet been adapted to handle cross-layer information.

5.4.4 *Benefits for Other Applications*

Beyond this work's central motifs of reliable streaming, other applications could benefit as well. First, cross-layer information could be utilized to directly improve the user interface and experience. Any information of upcoming service interruptions or other adverse conditions are simply displayed to the user. This is specifically helpful for interactive communication — as in video calls or VoIP. An unsuspecting user might be more startled at a sudden loss of reception than the one that was informed beforehand. The communication partner can additionally also be informed. While user hinting and notifications does not improve the actual QoS of the application it can still positively improve the perceived quality or QoE. But detailed user studies would need to further verify this.

Generally, any application that can locally exert control over its traffic and does not have to rely on server-side control will benefit the most. Also, the traffic should ideally be composed of smaller objects available in multiple versions and free to be reordered. A good candidate would be Web browsers with their stateless HTTP requests of a Web site, which is comprised of many small objects that can be, to a certain degree, requested in any order. Those objects could be requested in such an order to better accommodate network events announced by the lower layers. On the other hand, RTP-style streaming traffic might not be able to beneficially utilize cross-layer information as just a stream of continuous data is pushed to onto the client.

Directly involving the user itself are a further category of cross-layer interactions that are only available if there is a downward control flow through the broker to the radio layer protocols. This would require the additional presence of a user-space policy manager. The manager would offer the user a series of preferences and a configuration interface for a rule-based cross-layer control engine. With this the user could create complex compound policies such as: "Do not handover to a stationary WiFi from 3G when moving faster than $50\,\mathrm{km\,h^{-1}}$, switch only to in-vehicle WiFi if available." or "Avoid any vertical handover, which would interrupt my service for a long time, while a VoIP call is running."

5.4.5 *Implementation Outlook and Approaches*

As the model displayed, the target applications should not be directly (or indirectly through the inclusion of a third-party software library) responsible to retrieve cross-layer information. Instead, the broker was suggested as the means to implement cross-layer exchange in an actual software environment. This daemon — located in the user space — collects information from the lower layer network protocols which usually reside entirely in kernel space. The kernel typically exposes only some data publicly with a stable API and Application Binary Interface (ABI). Other data can often be derived from internal data structures which usually change with every version. The task of adapting to this changes and collecting concise data is handled by the daemon. Only the actual reactions to these information points must be implemented in the application itself.

The daemon provides all information through a pre-defined user space interface to interested parties. This includes the raw data itself, e.g., current latency or bandwidth information, as well as derived and predicted data points. Examples for the latter are the discussed early handover warning or mobility predictions. To further improve predicted values the daemon also integrates data from other device sensors outside the classical network stack, amongst others location and movement data and as well as system and battery state.

To further decouple the applications from the broker, the information should be provided through a shared Inter-Process Communication (IPC) bus. Current candidates for the reference implementation are Android, using the provided Intent[6] IPC mechanism, as well as Linux distributions using D-Bus[7]. An even higher level implementation might be possible for the latter case, as the D-Bus-based NetworkManager[8] framework already provides some of the network-related functionality. For example, it would be rather simple to implement switching the active network interface on the basis of cross-layer data with NetworkManager.

Through these decoupling efforts the broker's implementation and binary package can be completely swapped with another and all applications would still work as long as the bus interface remains the same. Therefore, the user and her chosen applications are not bound to a specific cross-layer broker provider with predictions conducted by certain algorithms.

6 https://developer.android.com/reference/android/content/Intent.html
7 http://www.freedesktop.org/wiki/Software/dbus/
8 https://wiki.gnome.org/Projects/NetworkManager

Rather, she can easily supplant the existing provider with a better one. The planned broker is therefore only meant as a reference implementation. Additionally, multiple brokers could even be simultaneously active as long as as long as their provided data does not intersect.

The viability of cross-layer information can be evaluated in several ways. A pure network-level simulation can give initial hints on performance gains and issues. But only the evaluation of data traces and packet level captures of an actual reference implementation might give good insights into the implications of diminishing the network stack's layer encapsulation properties. A further comparative statistical analysis of several different approaches to cross-layer data predictions algorithms as well as the specific application's reactions could also prove to be of significant interest. Both the testbed and LTE simulation approaches developed in Section 6.2 can help in validating this cross-layer approach.

5.5 Mobile Streaming Summary

To summarize, most application layer protocols are sensitive to circumstances of the lower protocol layers. Reliable streaming is no exception to this rule. Due to the time constraints present in media streaming it might even be more sensitive to outside influences than some other protocol with more relaxed timing constraints. Those protocol layer influences are aplenty, especially in mobile networks, and are most often unintended side effects of some protocol feature, behavior, or control loops that occur between protocols due to specific timings.

But this chapter showed that the mobile and Internet protocol stack is still very much in ongoing development and many of these side effects are known and being worked on. This happens either through the replacement of old protocols with entirely new versions that attempt to avoid these influences, or changes to existing protocols, as can be most prominently seen in the steady changes to TCP. However, as it might not be enough to improve the individual layers, a concept for cross-layer cooperation between protocols on different layers is also highly desirable. The framework provided here should serve as a blueprint for future reliable applications to correctly facilitate information from the mobile network, e.g., handover event information.

6

EVALUATING RELIABLE STREAMING IN MOBILE NETWORKS

Testing and evaluation plays a crucial role not just in correctly utilizing cross-layer data but also for media streaming in mobile networks in general. All of the discussed layer influences, interactions, and other unintentional side-effects need to be thoroughly understood for the intended goal of optimizing reliable streaming in mobile networks to be achieved.

The core network trace evaluations undertaken in Chapters 2 and 3 were already able to highlight the load influencing dynamics of the core network control plane. But many more aspects remain uninvestigated, some of which may not even be covered by this kind of passive network-wide measurement type.

To further the understanding of the behavior of reliable streaming in mobile networks and investigate influence sources, usually either active measurements or simulations are conducted. Both methods and their drawbacks and benefits are discussed in this chapter. Moreover, an approach to enhance active measurements for mobile devices that facilitates meta-data of the device and improve measurement accuracy, is presented in the first section. Afterwards, a full LTE-based reliable streaming simulation is also introduced and some initial results are given.

6.1 Active Measurements and Testbeds

Active measurements includes any kind of approach that generates its own network traffic and bases the evaluation on it. Therefore, they are usually conducted on an end-to-end basis, with at least one side under the control of the experimenter.

Apart from researchers writing their own specialized application for a singular study, active measurements are most often conducted with the help of existing network testbeds, thereby reducing the overall effort. Any new protocol or alterations to existing network protocols are usually best tested in large-scale network testbeds to collect performance data and find side-effects. The presence of background traffic and a large device heterogeneity and diversity are often considered an advantage to a testbed as it better resembles real networks.

Testbeds can be either constructed physically by setting up a number of dedicated machines in a lab or form a virtual overlay spanning over an existing computer network. Both of these principles work well for wired networks and there are a lot of examples of successful testbeds constructed with these principles. Currently, one of the largest installations is PlanetLab [Chu+03], consisting of dedicated machines located at a number of University sites. An experimenter can instantiate a *slice* — a portion of the overlay network made up of a number of interconnected VMs hosted on the various machines — and conduct her studies. Thus, many experiments can run concurrently on the same testbed without causing interference in the other experiments.

Several testbeds devote themselves to wireless and mobile networks. These are usually either assembled in a closed laboratory environment, SmartLab[1] or EmuLab[2] come to mind, or hand out subsidized mobile devices, prepared with custom firmwares, to volunteers which need to allow running network experiments on them. The approach is taken, e.g., by NetSense [Str+13] and PhoneLab[3] [Nan+13]. The latter approach raises some interesting issues concerning the volunteers' privacy which will be covered in the next section.

All of these testbeds consist of rather uniform nodes and access to it requires some amount of manual administrative effort. Other testbeds take an alternative route on these issues, for example the Seattle Testbed[4] [Cap09]. Its overlay network comprises of small software sandboxes installable on a wide range of device types. Experimenters gain automatic access to a portion of the overlay through a clearing house and are resource-restricted to their slice by a hypervisor. As Seattle is available to both conventional desktop and server computers as well as Android smartphones, the overlay composition very heterogeneous. Therefore, the node stability and availability can vary substantially over time, caused amongst other things by the individual uptime of the computers following typical diurnal cycles and mobility-related connectivity issues. While this results in a more realistic picture of a network, it simultaneously makes the execution of the actual study more difficult as these churning issues have to be dealt with.

The requirements specific to conducting reliable streaming experiments are low. As usually a fully pull-based approach is utilized, full control over the server containing the streaming files is not necessary. It is also generally not required to display the actual video on the client in reliable streaming as it always will be a pixel-perfect representation anyway and no-reference QoE metrics are usually employed. Therefore, the buffer-emulation measurement approach presented in Section 4.4 can be utilized here. The measurement devices only have to record the transmission traces of the streaming process and make them available to the emulation process. The latter can for example also be performed in a centralized and asynchronous matter as long as only non-adaptive streaming is measured.

6.1.1 *Enriching Mobile Measurements with Additional Metadata*

While transmission traces might be the bare minimum amount of data to conduct reliable streaming measurements, meaningful mobile measurement can include much more metadata, giving insightful indications of the device's current state.

Mobile devices have access to a vast array of data. On the one hand, the networking stack does itself contain much information on its state, as discussed in the previous chapter. The current radio and mobility state is especially relevant to mobile streaming measurements as it directly influences the link's QoS parameters.

1 http://smartlab.cs.ucy.ac.cy/
2 http://www.emulab.net/
3 https://www.phone-lab.org/
4 https://seattle.poly.edu/html/

But modern mobile devices additionally contain an ever-growing number of sensors, including GPS, accelerometers, temperature, pressure, luminosity, humidity, fingerprint, heartbeat, and several cameras and microphones. Moreover, further radio interfaces (WiFi, Bluetooth, NFC) gather data on network availability and signal quality. Each additional data source can help in quantifying the physical position, orientation and state of the device and even quantify the "state" of its user. Even surveillance agencies are allegedly more interested in metadata in general than actual data for this reason.[5]

Active measurements in mobile networks should make use of metadata and correlate it to the regular measurement data to achieve a finer result granularity. Alternatively, sensor data can also be used for new kinds of evaluations. Questions like "Can this device stream video at a satisfactory quality at certain locations?" can be raised and answered.

6.1.1.1 *Metadata and Privacy*

While the actual experiment usually runs in a strict sandboxed environment with no access to actual data on the device, allowing sensor and metadata access intentionally creates holes in this isolation model. As informative metadata can be for network studies as intrusive is unmediated access to this data for the device's user.

In a pure lab environment this poses no problem as there will be no device owner whose information can be leaked. However, preserving the privacy of actual device users running network testbed software is absolutely critical for a number of reasons:

- For ethical reasons as stated in several ethics proclamations, for example in the well-known hacker ethics[6] which states to *"Use public data, protect private data."*. This is often called the principle of data protection and minimalization, aiming to prevent abuse and minimize the effects of accidental data leakage.
- User acceptance is a very important aspect. Testbeds need to have a sufficiently large installation base to achieve meaningful results. But users might choose not to participate if the testbed application is too intrusive. There needs to be a balance between revealing and protecting privacy sensitive data. The acceptance can also be increased by information and disclosing employed data. This serves the additional purpose of educating the user of privacy sensitive data.
- Lastly, there are legal constraints to consider when dealing with personal information. In Germany, for example, several fundamental constitutional rights restrict the use of personal data. These are the *"Recht auf informationelle Selbstbestimmung"*[7] and the *"Grundrecht auf Gewährleistung der Vertraulichkeit und Integrität informationstechnischer Systeme"*[8].

5 http://www.wired.com/2013/06/phew-it-was-just-metadata-not-think-again/
6 http://www.ccc.de/hackerethics
7 https://www.bmi.bund.de/DE/Themen/Gesellschaft-Verfassung/Datenschutz/Informationelle-Selbstbestimmung/informationelle-selbstbestimmung_node.html
8 https://www.bundesverfassungsgericht.de/entscheidungen/rs20080227_1bvr037007.html

To conclude, network measurement testbeds have to regulate, filter, adjust, or outright prevent access to sensitive data. Existing testbeds usually do not take a technical but rather a bureaucratic approach to this problem. For example, all experiments in the aforementioned PhoneLab require approval from a review board.

But that does not mean that there are no technical means available. Looking outside of the field of network testbeds, some research and even usable implementations are available.

A very basic and static approach to protecting privacy data is implemented in the regular versions of both iOS and Android. During installation any application has to request specific rights for each sensor source to be able to use it in the future. The "Privacy Guard" feature[9] in the custom Android firmware CyanogenMod[10] provides the additional capability to monitor and revoke any sensor permissions at runtime. Other approaches include increasing the sensor permission granularity [Jeo+12] or tagging sensor data at the source to better track and find privacy leaks in applications [Enc+14].

6.1.1.2 *Sensorium Framework*

Even with these approaches at hand, the privacy issue should be mitigated further beyond the binary allow/deny access approach. Also, it usually is a complicated task for experimenters to access all the various sensors,each answering to a different API. *Sensorium* [RFP13], an application for the Android OS, was created during the course of this thesis to remedy both of these issues. Sensorium is a generic unified sensor reading framework that interfaces with other applications as well as network nodes and provides the sensor readout to them. More importantly, a fine-grained multi-level privacy control model is implemented.

Architecture

Figure 6.1 overviews the components of Sensorium's architecture. The individual sensor drivers are placed on top of the operating system and take care of reading sensor values from platform-specific interfaces and pushing them upwards into a common registry to be read by the sensor reading API. All sensor data is relayed both to a local display as well as to all outbound interfaces. Due to the layered architecture, contributors can simply extend or substitute the existing layers. New drivers or additional outbound interfaces can be easily implemented in this way.

The existing outbound interfaces include an HTTPS client that pushes JSON data to a configurable network experiment server where the data can be further processed. To connect to programs running locally on the same device, the XML-RPC interface can be polled. Pull-based access from sources other than the local host is not allowed for security reasons. However, any data leaving Sensorium must first pass through the privacy layer, which filters the data based on the user's preferences.

9 https://plus.google.com/+CyanogenMod/posts/gk7X3HjNvnH
10 http://cyanogenmod.org/

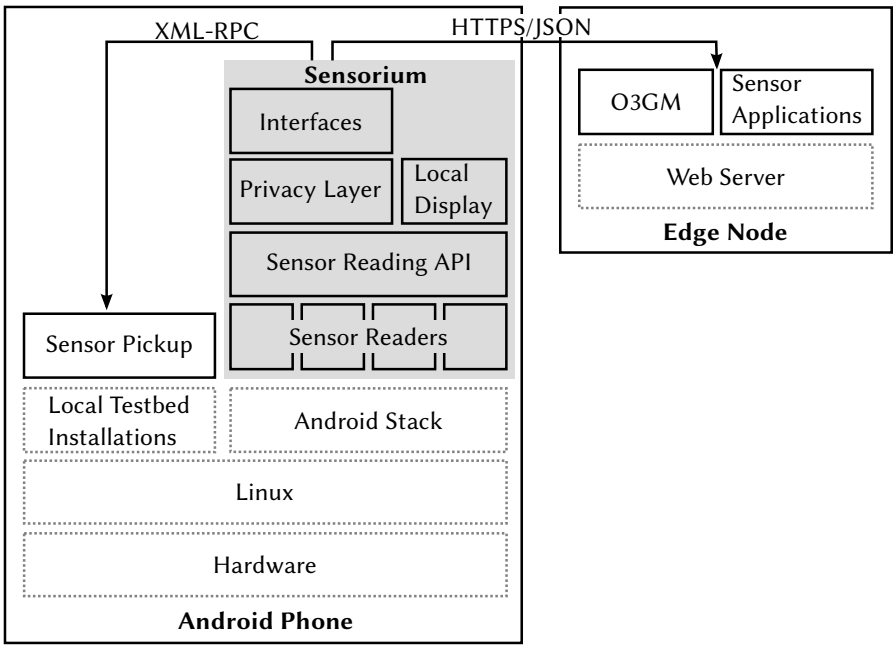


Figure 6.1: Sensorium architecture interfacing with other applications. Previously existing components are marked with a dotted line.

Privacy Model

This privacy layer allows for a fine-grained control over the amount and precision of data that the system is sharing. A settings GUI provides the user full control over her privacy. Here, she can choose the amount of information she is willing to reveal for each sensor. Based on this preference, the privacy layer then either outright prevents access to specific values or reduces their impact by anonymizing them.

This anonymization task can work in several levels. Apart from completely replacing it with a running number, the value could also be hashed. In many cases context-sensitive anonymization approaches are available. For example, the precision of a geo-location could simply be reduced down to a city-level or even country-level.

As the user is always in firm control of the process, she should be more accepting of actual network experiments that use this kind of data. However, such studies need also cope with the fact that they might get incomplete or partially anonymized data.

Implementation

Sensorium is currently targeted at and implemented on the Android platform.[11] It can be installed on any Android system without modification. Screenshots of the sensor display and

11 More information is available at https://github.com/fmetzger/android-sensorium.

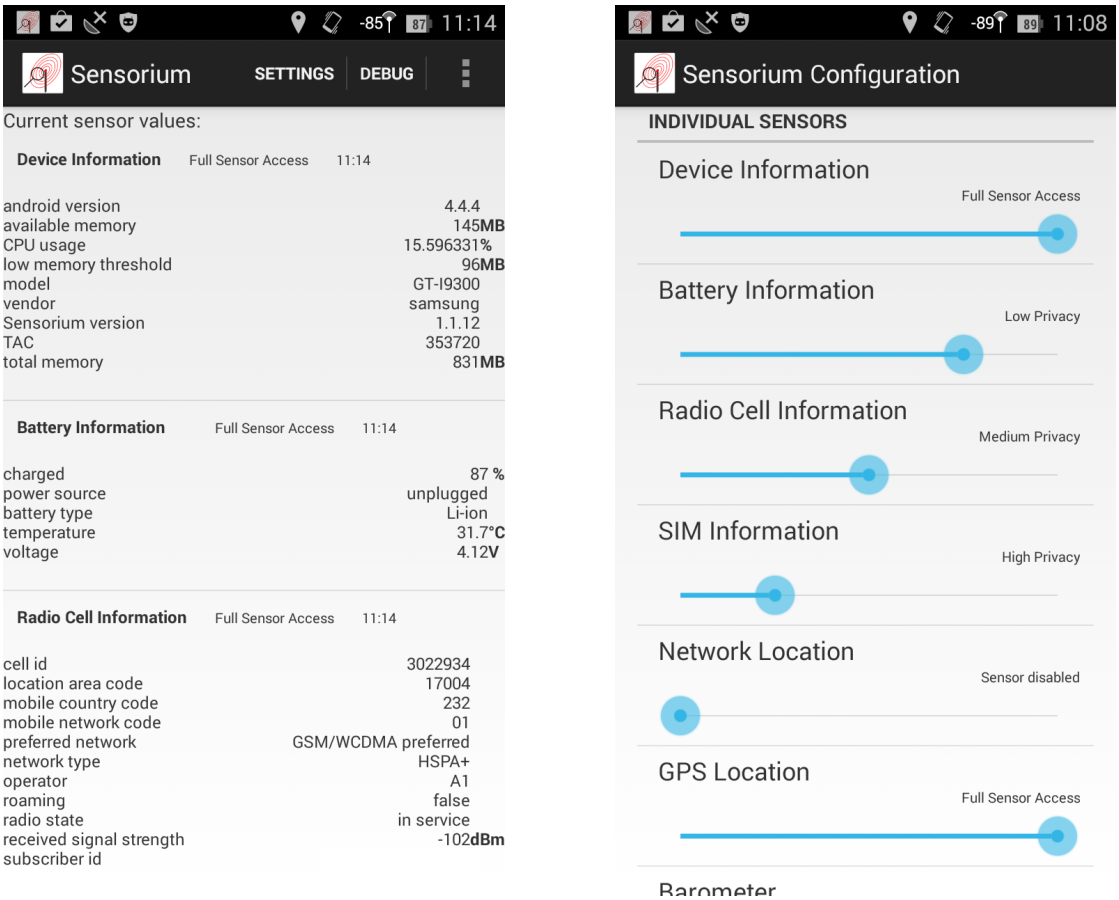


Figure 6.2: Sensorium sensor values display and settings screenshots.

settings menu of the application are depicted in Figure 6.2. Drivers for most of the typically available sensors are implemented.

Sensorium was demoed at the NetSys 2013 conference and has since then been available in various app stores. Data from the Google Play Store suggests at least 50 concurrently running device installations across several countries, carriers and device types. Albeit a still low number, as there was no real advertisement conducted, this could indicate its feasibility as a companion application to a network testbed. Of note are also two related applications that intend to follow up on Sensorium's approach in the future: The Sensibility Testbed[12], which tightly couples itself to the aforementioned Seattle platform, and BlurSense[13] [Cap+14], aiming to improve the sensor value anonymization efforts.

12 https://sensibilitytestbed.com/ and https://github.com/SensibilityTestbed
13 https://blursense.poly.edu/

Sensorium Usage

Sensorium was originally written for two specific targets. First, it was intended as a prototype sensor extension to the Seattle testbed. Pre-anonymized data was to be delivered through the XML-RPC interface to a locally running Seattle installation. Any experiment running in this sandbox could then facilitate this sensor data.

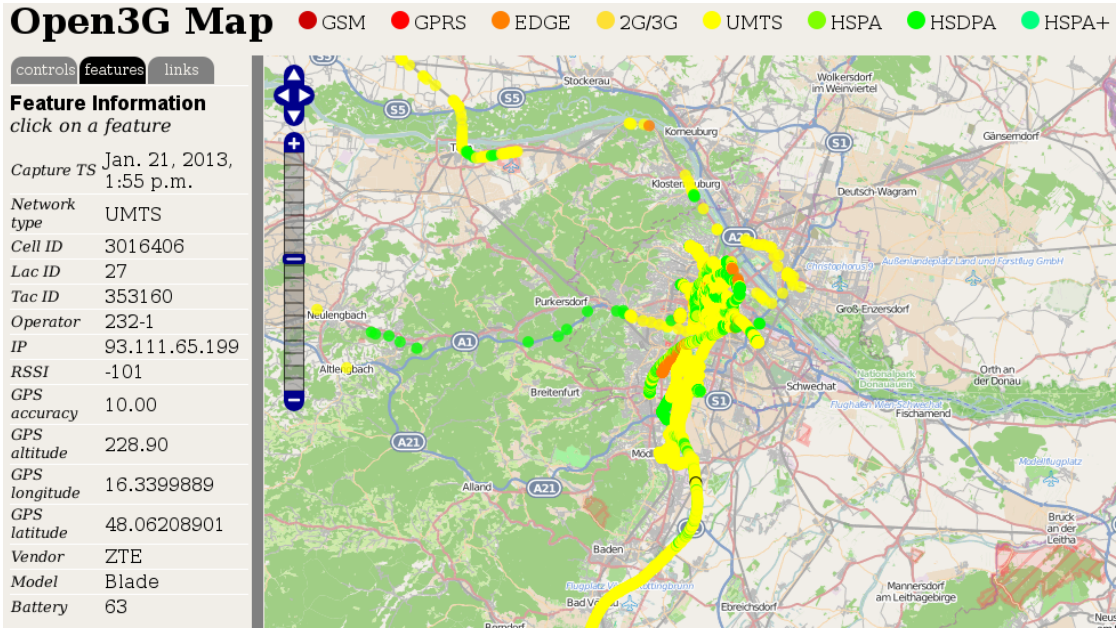


Figure 6.3: The O3GM web page, displaying a 3G coverage measurements layer with data collected by Sensorium on top of the OpenStreetMap base layer.

Second, to present what actually can be done with smartphone sensing data, Open3G Map (O3GM)[14][15] was created, which visualizes mobile coverage data. O3GM is a Web service, that makes use of Sensorium's JSON upload feature, displaying geo-located 3G radio access quality information in an OpenStreetMap overlay (using OpenLayers[16]) as can be seen in Figure 6.3

Coverage data is usually only available to mobile operators, which have no interest in sharing this information. Other projects such as OpenSignalMaps[17] and Sensorly[18], as well as corporations like Google and Apple collect these data, but are very restrictive regarding usage by third parties. O3GM data is freely available under an open content license.

Coverage data can again also be helpful to determine the viability of mobile reliable streaming. For example, knowing the RAT and signal strength of a specific path can help make informed

14 https://o3gm.cs.univie.ac.at/o3gm/
15 https://github.com/lukpueh/Open3GMap
16 http://openlayers.org/
17 http://opensignal.com/
18 http://www.sensorly.com/

decisions on which stream quality to choose at which time for adaptive streaming. If Sensorium would be combined with the reliable streaming measurement framework of Section 4.4, one could check rather easily for correlations between stalling phases and mobility events.

6.2 Mobile Reliable Streaming Simulations

Some experiments cannot be easily conducted in active measurement testbeds, for example due to the necessary scale to achieve meaningful results. Often new protocols or adaptations to existing ones are preferentially first tested using a network simulation. Simulation frameworks are especially important for mobile networks, as acquiring packet-level traces and information about every node in an actual commercially operating cellular network is nigh impossible due to the users' privacy and the provider's business concerns.

While there are some active measurement studies specifically targeted at reliable streaming in mobile networks, e.g., [MLT12], most are conducted either in fixed networks or using simulation frameworks. Therefore, to better evaluate reliable streaming, it would be very desirable to find an existing framework that covers all important aspect in 3G or 4G mobile networks, including the influences of the core network and control plane signaling.

There are a number of network simulators readily available for use, both commercial as well as FOSS. But only a small subset of them has the capability (or can be extended) to simulate 3G or LTE networks. Even further limiting is the circumstance that most radio network capable simulators only concern themselves with the physical radio link and completely neglect all other network paths, especially the core and all control plane signaling interactions. The following list overviews current publicly available simulation frameworks with 3G/LTE support:

- An external UMTS module[19] is available for the no longer maintained **ns-2** network simulator. A further, separate collection of patches[20] also extends ns-2 with UMTS radio link capabilities [VŠR11] but it is not publicly available. Both extensions are, as of August 2014, no longer being developed and not up-to-date to the newest 3GPP specifications. They also focus solely on the user plane radio physical and link layer of UMTS.
- Another third-party radio link layer simulation model is available for **MATLAB**[21] [Meh+11].
- A standalone LTE simulation[22] [Pir+11] includes models for some LTE nodes, including the Evolved Node B (eNB) and MME and implements a selection of protocols (PDCP, RLC, and RRC). However, the implementation of these nodes and protocols is rudimentary at best and is not even close to the actual specification. Additionally, the simulator lacks a coherent TCP/IP stack as IP is reduced to its basic functionality and TCP is completely absent.

19 http://net.infocom.uniroma1.it/reti_files/reti_downloads.htm
20 http://seacorn.cs.ucy.ac.cy/eumtssim/
21 http://www.nt.tuwien.ac.at/research/mobile-communications/lte-simulators/
22 http://telematics.poliba.it/index.php/en/lte-sim

- A framework dubbed SimuLTE[23] is available for **OMNeT++**[24]. Included are the user plane aspects of the radio link and some basic SGW and PGW functionality.

- The **ns-3**[25] simulator already contains an LTE/EPC module called LENA[26] [Bal+13] with features similar to SimuLTE. Again, only user plane SGW/PGW functionality is present with an initial GTP-U implementation.

The goal here is to simulate reliable streaming in a realistic mobile environment. That would ideally include both a complete horizontal network path — both the radio link as well as the core network — as well as vertical network stack — comprising both user plane and control plane. Unfortunately, none of the above feature a complete representation, which can be at least partially attributed to the complexity of the 3GPP specifications. Nonetheless, the simulators could still provide a viable basis for a mobile streaming framework while keeping the limitations in mind. Between these a decision needs to be made as to the basis of the mobile streaming simulation framework.

Ultimately, the choice as a basis for reliable streaming simulations fell on ns-3 with the LENA module. Alongside with SimuLTE it has the most complete LTE representation. And targeting LTE networks seems to be the more future-proof path in the long term. With the exception of OMNeT++, which has comparable capabilities, ns-3's TCP/IP is much more complete and realistic than that of the other frameworks. Additionally, it can also incorporate the actual TCP/IP of older Linux kernels with NSC[27]. As an additional feature, ns-3 can also act as a network emulator for real network traffic. This can be exploited through the upcoming model.

In the long run, to better represent actual mobile networks the base radio framework in ns-3 would need to be extended with more control plane aspects — which are unfortunately almost completely absent in any current simulator — and adapted to the latest 3GPP specifications.

6.2.1 *Simulating Mobile Reliable Streaming in ns-3*

With ns-3 chosen and the core network model set, the task is now to define and simulate reliable streaming on top of the LTE network. To properly evaluate the setup a number of measurement series also has to be defined and conducted.

Through this simulation testbed, arbitrary reliable streaming playback models can now be tested and optimized for the various conditions and pitfalls of mobile networks. Of special interest could be the relation to mobility phenomenons and issues occurring during handover.

23 https://github.com/inet-framework/simulte
24 http://www.omnetpp.org/
25 http://www.nsnam.org
26 http://networks.cttc.es/mobile-networks/software-tools/lena/
27 http://research.wand.net.nz/software/nsc.php

6.2.1.1 *Simulation and Emulation Setups*

To begin, the model behind the progressive streaming measurement framework based on the playback buffer employed in Section 4.4 can be reused in the simulator. But instead of being an off-line analysis algorithm for network traces it is used as an actual emulated streaming client here.

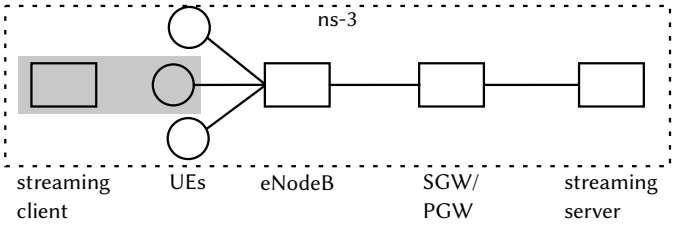


Figure 6.4: LTE reliable streaming simulation testbed.

The rest of the network model is kept as simple as possible as is apparent in Figure 6.4. LENA readily provides (incomplete) implementations for the UEs, eNB, and a combined SGW/PGW node. For the streaming simulation two things were added into this network: a streaming server and a streaming receiver and player.

For the streaming server a node from which the video segments are pulled was connected to the Internet-facing link of the PGW. The storage server is kept as simple as possible. Upon receipt of a request for a specific segment over TCP a dummy segment with the correct size is immediately sent to the client, reusing the open TCP connection. The TCP congestion avoidance mechanism implemented by ns-3 is New Reno, which also might not be ideally suited to mobile environments. Especially when compared to newer congestion avoidance mechanisms that are optimized for a high BDP like Linux's CUBIC.

In this setup TCP will be directly used to transport the stream instead of for example the more commonly used application layer protocol HTTP. This should make no difference on the streaming process in the long run apart from the absence of a slight protocol overhead (i.e., the HTTP headers) at the beginning of the transmission and for each segment. Although some features coming with HTTP/2 cannot be easily tested in this way, especially the application layer multiplexing aspects. However, using TCP directly simplifies the simulation and makes the results easier to interpret.

The client's playback buffering model is implemented as an application running on the UE. The device also initiates the streaming and takes care of requesting each individual segment. For the purpose of this simulation the segments do not contain actual video data. Rather, the playback simulator just reads the list of frames and their size and synchronizes this information with the received amount of data to correctly calculate the buffer level.

During transmission, the existing LTE framework sets up the lower layer protocols, which includes the radio stack and the GTP core network bearer, accordingly. In the simplest model, only one eNB and one UE are present, to avoid interference with other devices. Additionally, a mobility model has to be selected for the UE, on which the streaming client is running, in

order to conduct the experiments. The simplest case would be a stationary mobility model at a close distance to the transceiver, laying more emphasis on testing the influence of the overall network path rather than mobility factors. This should also give a measure of the upper limit of the achievable reliable streaming performance, all other mobility models will undoubtedly reduce the performance.

The scenario can now be easily extended to factor in multiple devices and an actual mobility model with handovers. The necessary capabilities to set this up are already present in ns-3's LTE model. Apart from the described settings the LTE nodes are otherwise left at their default configuration, which should yield an overall net bandwidth of 80 Mbit s^{-1} in the LTE radio cell.

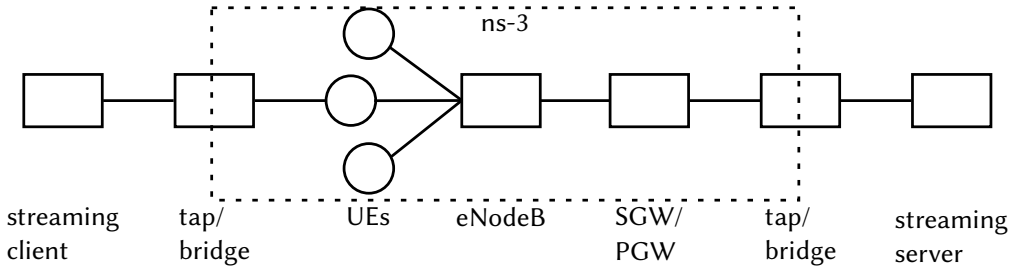


Figure 6.5: Future testbed iteration: hybrid of ns-3 LTE simulation and actual or emulated streaming client and server bridged to it.

Beyond its application as a pure simulation, a simulated ns-3 LTE network could additionally be facilitated as a network emulator. Figure 6.5 demonstrates such a setup. To be able to use it for a streaming emulation in the style of the measurements conducted in Section 4.4 a bridging device would need to be added on each side of the network that interfaces with a real testbed network. The only task of the simulated part is then to alter the QoS parameters of the link according to its model. This approach is ideally suited to quickly test existing and already implemented streaming solutions for use with a mobile network and saves the effort of reimplementing them as a simulation model.

6.2.1.2 *Simulated Streaming Models*

Following the classifications and playback models from Sections 4.1 and 4.3.2 the simulated model is a pull-based segmented video streaming system using TCP. Two streaming models are suggested here, with the first suited for plain reliable streaming and the second modified to utilize the benefits of adaptive streaming. Other playback models could be easily implemented as well, but these two should closely resemble the usual approaches taken by real reliable streaming players as was previously already observed.

Four Threshold Segmented Streaming Strategy

The first playback strategy is governed by four thresholds. They are, ordered according to the buffer level they represent, from lowest to highest:

1. *Playback stop.* Represents the lower limit of the buffer and the player will stop if the level goes below this threshold. It can be set to as low as 0 s but it defaults to 0.5 s for this model. When the limit is underrun, stalling will occur.

2. *Transmission start.* If the transmission is currently stopped due to reaching the buffer limit, restart it if it falls beyond this threshold. Naturally, at the start of the streaming process segments are already requested at a buffer level of zero. The default is set to 2.5 s of video in the buffer. The gap between this and the playback stop threshold should be large enough to avoid stalls because of transmission restarts occurring too late.

3. *Playback start.* This threshold takes effect after every stalling event and restarts the player if at least this amount is in the buffer. It is set to 5 s here.

4. *Transmission stop.* In some scenarios this last threshold might not be necessary as it only acts to prevent a buffer overrun and lost video data. But most mobile devices suffer from rather low memory constraints and therefore need to limit the video buffer. When this limit is reached, no new segments are requested until the buffer falls below the transmission start limit again. It is set to 10 s here and is a soft limit that can be exceeded by already requested segments that have yet to arrive. Therefore, it always should be set slightly below the buffer's actual hard limit.

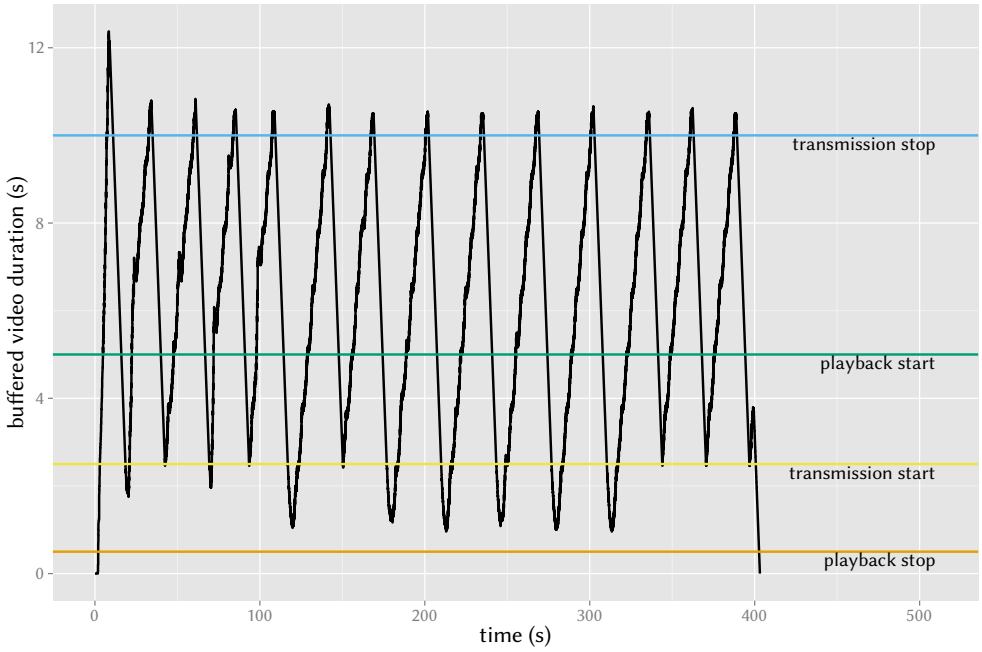


Figure 6.6: Sample simulation run demonstrating the four threshold strategy.

To demonstrate the strategy an example scenario is displayed in Figure 6.6. Depending on the relation of TCP goodput to video bitrate a player with this strategy should always bounce

between the transmission start and stop thresholds and never stop playing intermittently as in the example. The default threshold values given here may be far from ideal and need to be optimized in the long run. But this is often the same case in real streaming solutions with the players arbitrarily preconfigured to values that could make sense from the developer's point of view without scientific validation.

Six Threshold Window Scaling Adaptive Streaming Strategy

By adding two additional thresholds the strategy could also be extended to work for adaptive reliable streaming. As a further prerequisite, all video segments need to be available in a number of quality levels and therefore video bitrates and seamless switching between levels at the segment borders needs to be possible. The more quality levels are present the better adaptable the streaming client can be. It can already work with as low as two levels, but this coarse switching between a very low number of levels will also be more noticeable by the user and might disturb her QoE more than a smooth decline in image quality.

The two new thresholds are values to trigger the switch to the next higher and lower segment quality level respectively. The reduce to lower quality threshold should be located between transmission start and playback stop, the increase to higher quality situated between playback start and transmission stop. To accommodate more than two levels these two thresholds can be triggered multiple times. If the buffer level is still rising after the segment quality has been raised the quality should be raised again at the next opportunity.

To better find a stable segment level candidate, the player should compute and keep track of the incline of the buffer level in the last time window. The steeper it gets the higher the chance for a quality level change should be. The same rule applies also to the lower quality threshold. Any adaptive streaming strategy should also avoid flapping back and forth between quality levels. This is usually conducted through some kind of hysteresis.

6.2.1.3 *Scenario Evaluation*

To test the basic viability of the streaming simulation model (the validity LTE model is taken as-is and covered by other research [Bal+13]) several test scenarios are defined and subsequently evaluated. Common to them is the configuration of the video streaming simulation module. In every case the simpler four threshold strategy is facilitated and tested with the three videos described in Table 6.1, each representing a different quality profile. Only one simulation run was conducted for each parameter setting as there are no known random factors involved that would merit running with different random seeds.

Congested Server Link

This first series of evaluations exclusively looks at the QoS of the streaming server's link and leaves the LTE configuration unaltered. The radio subsystem is configured to have a stationary UE in close vicinity of the single eNB. It should therefore reach the maximum

Table 6.1: Parameters of the videos used in the streaming simulation scenarios.

Parameter	Low Quality	Standard Quality	High Quality
Video duration	318 s	602 s	596 s
Size	19.2 MiB	258 MiB	853 MiB
Frame rate	29.97 s^{-1}	29.97 s^{-1}	24 s^{-1}
Average video bitrate	504 kbit s^{-1}	3596 kbit s^{-1}	12 Mbit s^{-1}
Codec	H.263	MPEG-4	MPEG-4

attainable performance of the radio link as it will be undisturbed by other users and fading effects. Therefore, it also represents the best case scenario for mobile streaming. If the streaming strategies and the associated video files reveal issues under these conditions they will surely also fail in an environment with a more degraded radio link.

The QoS variables altered here are the link's bandwidth and delay. Loss will remain unchanged and at zero, as it will also result in just another source of delay for the reliable streaming process due to retransmission. Similar to the evaluations with the measurement framework of Section 4.4 the performance will be measured on the basis of the player's stalling characteristics, i.e., the total duration and the number of stalling phases. With this, more complex QoE metrics can then be calculated.

The first series solely alters the bandwidth of the streaming server's link up to a maximum of 1000 Mbit s^{-1}. The results are depicted in the Figures 6.7 and 6.8.

As soon as the link's bandwidth exceeds the video's bit rate, the number and duration of stalls are reduced. Stall events and duration in both the low and standard quality videos drop to zero. Only the high quality video does not show the same results. As its bit rate of 12 Mbit s^{-1} should be perfectly manageable for the radio link some other explanation is required. A possible hypothesis are issues with the approach to requesting subsequent segments. As these need to be requested in a timely manner while other segments are still being transmitted, they could be requested too late and do not saturate the link anymore.

However, this effect is somewhat contrary to the additional effect observed at link speeds of 100 Mbit s^{-1} and above, where the stalling phases suddenly see an unexpected increase. As this happens exactly when the transmission speed exceeds the radio links capacity of about 80 Mbit s^{-1} it might be an indication of the negative interaction of LTE's loss and congestion concealment with TCP's congestion avoidance mechanisms, related to the previously discussed bufferbloat issues. TCP can not properly detect the bottleneck capacity in a timely manner as LTE attempts to buffer and retransmit any packet exceeding the radio capacity below the user IP layer. Through this issue, the effective goodput seems to get reduced to about 5 Mbit s^{-1}, which is not enough for the high quality video to stream without interruption.

Once again utilizing Equation 4.6 as a validated QoE metric and applying it to the results of the bandwidth series, it is evident that the perceived quality really suffers in this scenario. The

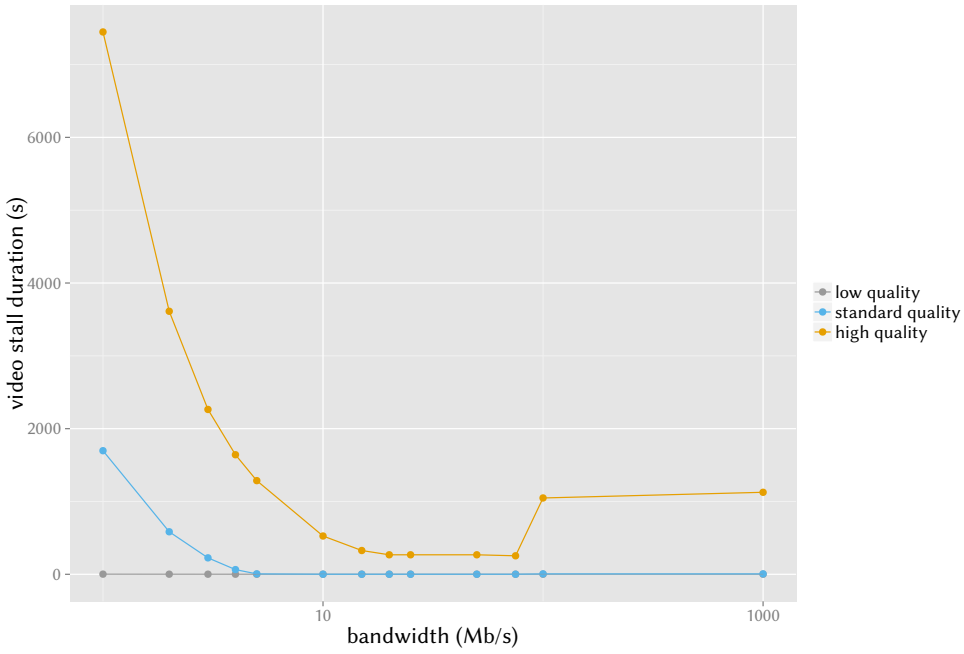


Figure 6.7: Relative stalling duration of the simulated reliable streaming player under limited Internet bandwidth.

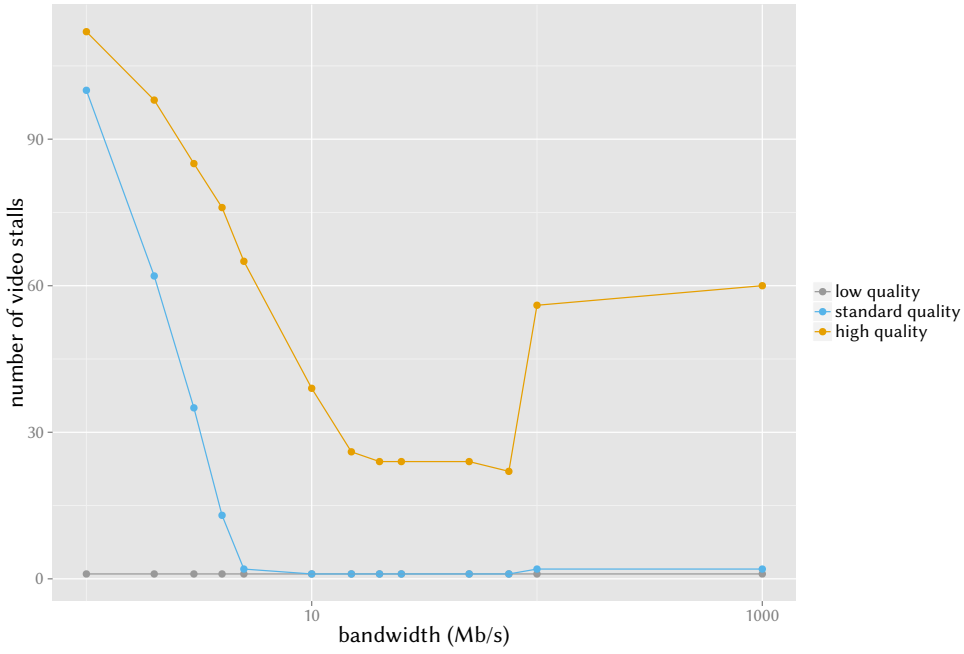


Figure 6.8: Number of stalling events of the simulated reliable streaming player under limited Internet bandwidth.

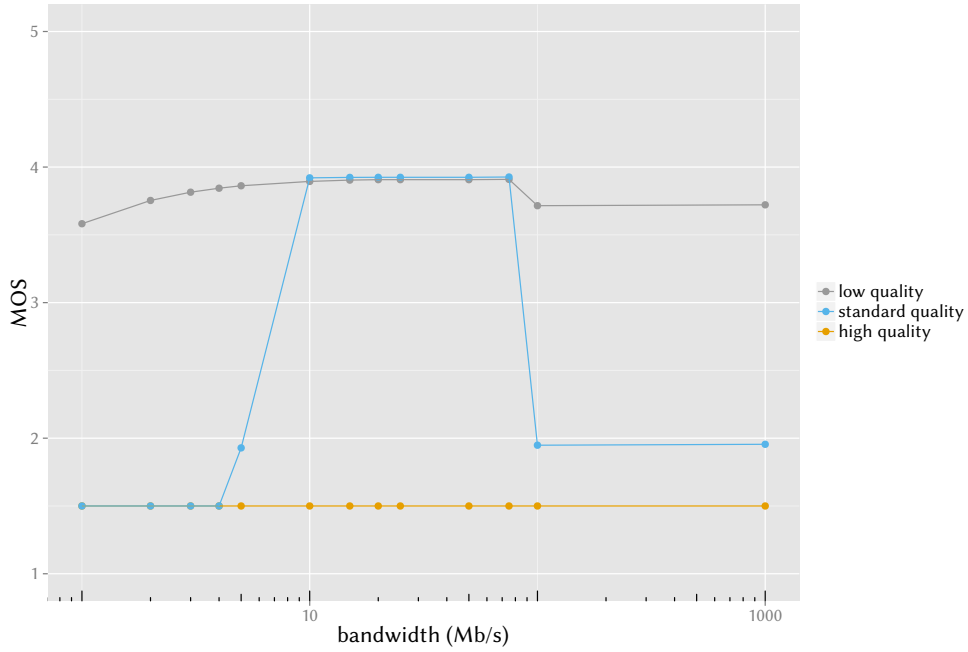


Figure 6.9: Computed QoE of the reliable streaming strategy with limited bandwidth.

MOS of the high quality video never exceeds the equation's base value of 1.5 and is effectively unwatchable according to the metric. The other two video profiles manage to achieve a good quality when sufficient bandwidth is present, with the issues above 80 Mbit s^{-1} in mind.

In a second series of simulation runs the latency of the streaming server's link was increased incrementally up to an additional latency of 1000 ms with the link's bandwidth fixed to 1 Gbit s^{-1}. The values were set deterministically with no probability distribution. Figures 6.10 and 6.11 again show the results in terms of the relative duration and number of stalling phases. The playback simulation seems to be excessively sensitive to latency increases. Even a small 100 ms increase makes the high and standard quality videos completely unwatchable as the stalling duration is more than ten times longer than the actual video. With latencies beyond 100 ms the high quality video could not be successfully streamed anymore. The computed QoE of this scenario is not displayed as it essentially never deviates from the base MOS of 1.5, verifying the detrimental impact of the latency.

This behavior can partially be attributed to the simplistic segment request strategy used in this experiment. New segment are only requested when the previous one has fully arrived. As this takes a full round trip, the latency has a significant influence on the arrival time of subsequent segments. To avoid this kind of stop-and-wait behavior in the segment request process, new segments need to be requested sufficiently in advance while the previous segment is still being transmitted, so that the full bandwidth is always utilized. With these improvements to the retrieval strategy this specific issue can be eliminated. But this just shows one of the

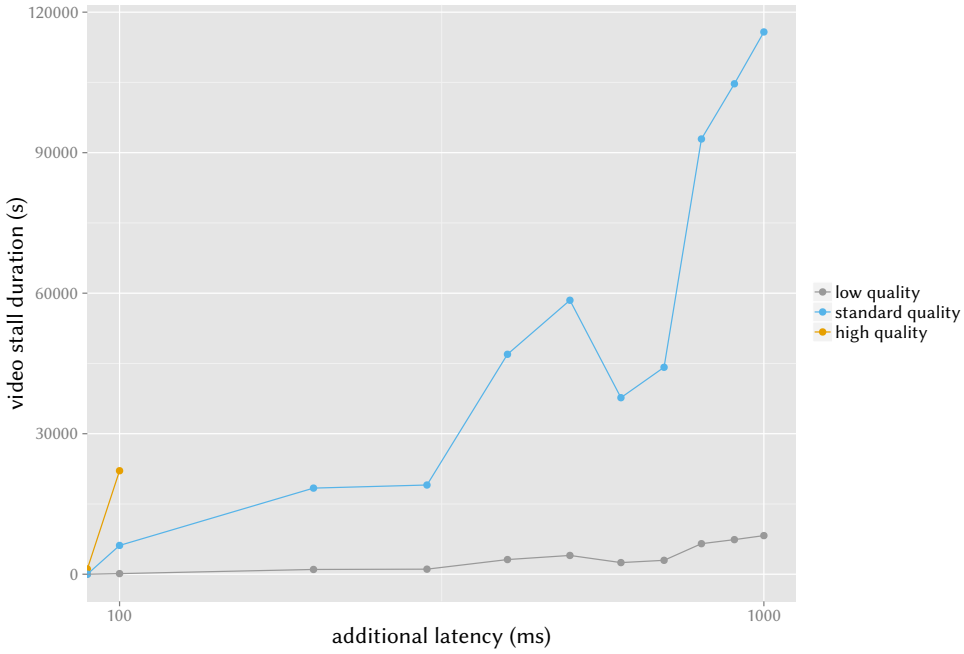


Figure 6.10: Relative stalling duration of the simulated reliable streaming player under increased Internet latency.

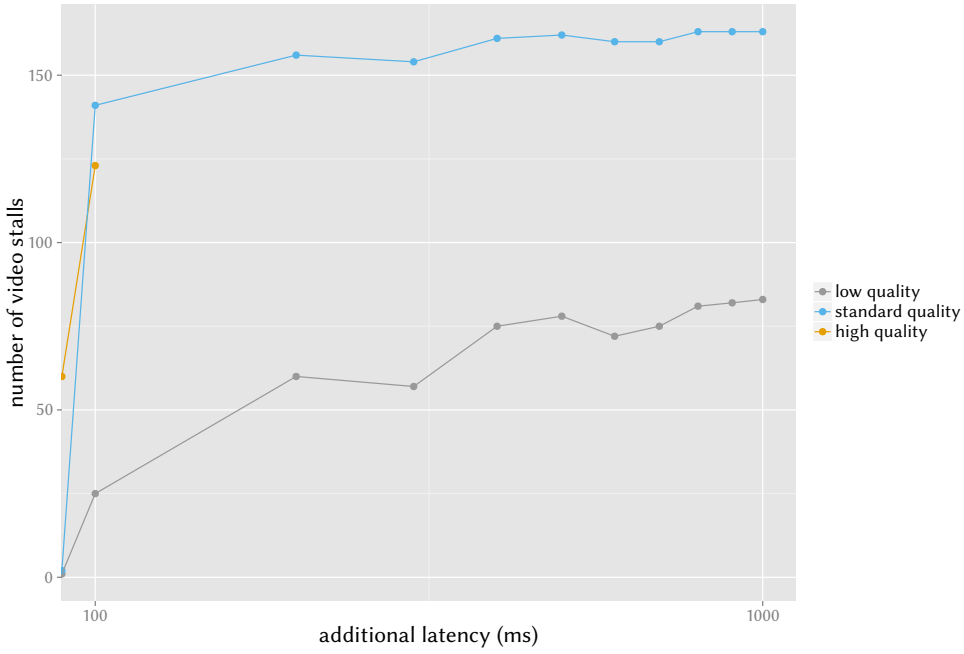


Figure 6.11: Number of stalling events of the simulated reliable streaming player under increased Internet latency.

many pitfalls and influences of various layers. A simple implementation detail may have large implications on the streaming quality the user experiences.

Device Mobility

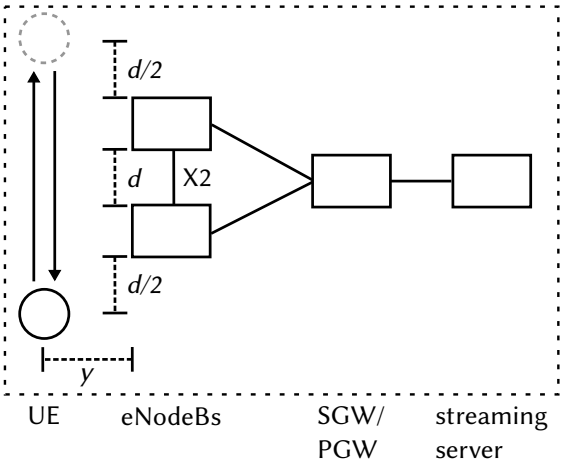


Figure 6.12: Simulated handover mobility scenario using waypoints.

Besides altering the server link, the effects of the simulated LTE network can also be investigated through various other means. In the following scenario mobility will be under scrutiny. For this, a second eNB will be added to the network. Both are interconnected through the X2 interface, which will be used for the eNB-anchored mobility provided by ns-3's LTE model. Instead of having a constant position relative to the eNB, the UE will now move back and forth between two waypoints on a two-dimensional plane in parallel to the two base stations which are placed at a distance of d between each other. A handover is triggered each time the device leaves the range of one station and enters the other. Furthermore, the horizontal distance on the 2D-plane between the device and the eNBs is controlled by a second parameter y. Figure 6.12 depicts this scenario and the positioning of the waypoints.

For the purpose of this experiment, the two eNBs were placed at a vertical distance of $d = 500\,\mathrm{m}$ while the horizontal distance to the device was varied between $y = 0\,\mathrm{m}$ and $y = 150\,\mathrm{m}$. The device is moving at a constant velocity of $20\,\mathrm{m\,s^{-1}}$. During the movement the standard quality video was streamed to the device and its playback simulated. The results in terms of the buffer fill level are displayed in Figure 6.13. The color-coded horizontal lines once again demark the same thresholds from the previously described four threshold segmented streaming strategy.

The handover events between the two eNB are denoted by the gray vertical lines and occur in regular intervals. The actual control transfer usually last for only about 40 ms. But looking at the figures it is immediately evident that the video buffer, and therefore the throughput of the video segments, already suffers seconds before this handover event when the edge of the

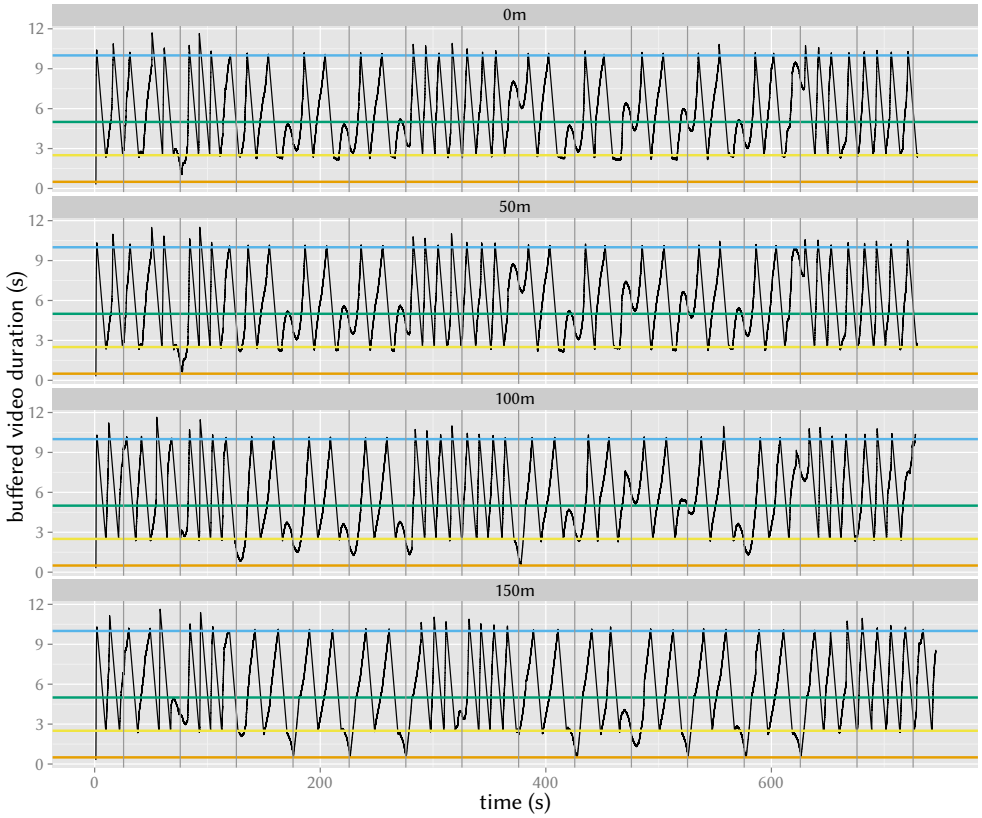


Figure 6.13: Playback buffer time series of the simulated mobility experiments with increasing distance between device and eNBs.

radio range of the currently active eNB is reached. The playback stop threshold is undercut several times right at the handover mark resulting in a playback stalling phase.

It seems that the throughput at the radio edge drops below the video's average bitrate of $3596\,\mathrm{kbit\,s^{-1}}$. Therefore, the buffer cannot be maintained for this quality level. If the buffer was at a critically low level beforehand, it will run empty and lead to stalling due to the mobility. Additionally, there seems to be a dependency between the stalling probability and the distance between device and base station. A farther distance decreases the achievable throughput leading to an increased chance of the buffer running empty.

Mobile streaming playback strategies can attempt to counteract mobility-related issues in a number of ways. A simple approach would be to adjust the threshold levels to allow for a much larger buffer while also restarting transmissions earlier. But this comes at the price of an increased memory resource usage on the device, which might not possible in every scenario. Second, the strategy can be exchanged for an adaptive one, for example the described six threshold window scaling strategy with appropriately chosen threshold values. Here, the stalls are exchanged for phases of lower quality video, which can be acceptable under circumstances.

Finally, both non-adaptive and adaptive strategies can be improved by employing the cross-layer model described in Section 5.4. By utilizing information from the radio network layers, handover events can be detected sufficiently in advance and an appropriate reaction chosen by the playback strategy. For example by temporarily increasing the maximum buffer threshold and requesting segments more rapidly or by quickly dropping to a lower quality level and filling the buffer up.

Further Scenarios

Apart from the experiments conducted here, some others scenarios listed here are also worth investigating but out of scope for this work.

If one plans to implement a reliable streaming solution, the simulation can be utilized to test any kind of playback strategy and transmission mechanics and attune it to mobile networks. Besides developing entirely new playback strategies, the default thresholds of the four and six threshold strategies should also be optimized through the iterative simulation of value combinations.

The opposite approach would also be possible for mobile network operators: The operator's mobile network infrastructure can be mapped onto the simulation and the performance of existing streaming strategies inside this network can be tested. If any issues or performance bottlenecks are discovered, the operator can make adjustments to the simulated infrastructure and implement these changes afterwards outside of the simulation.

It would also be worthwhile to see the impact of the cross-layer model described in Section 5.4 as they are specifically intended to reduce the influence this kind of mobile network architecture has on user applications. An implementation of cross-layer hinting would be relatively simple here, as the layer isolation boundaries are much easier to circumvent in a network simulator and information on, e.g., handover events is readily available.

As mentioned earlier, ns-3's LTE framework does only implemented a limited subset of the 3GPP specifications, mainly related to the user plane and the radio link stack. Almost no control plane procedures are implemented. This may have only a negligible impact as soon as a connection is established and remains stable and only a very small number of users are connected to the system. But this is not the norm for an actual mobile network in operation with its mobility features and high churn, both in terms of data flows as well as mobile users, generating a large amount of signaling traffic as was discussed before in Section 2.1. To best mimic the dataset evaluations related to PDP context and GTP tunnel life cycle management conducted in Section 3.1, the simulator would need to at least be able to reproduce these tunnel management and the related RRC PDP context procedures happening on the user traffic path. Therefore, it is worth to reconduct the above experiments as soon as the framework has made any progress in this direction.

6.3 Summary

It is safe to assume that reliable video streaming in mobile networks can be evaluated in a large number of ways. Besides the previously taken road of passively recorded network-scale traces, two other approaches are most valuable in investigating reliable streaming, active measurements and simulations. Both bring along their own set of benefits and drawbacks.

Active measurements are better suited to test the wide range of influence factors present in actual networks and device protocol stacks. These can usually not be properly represented in a simulation. In order to increase the viability for mobile measurements, Sensorium was introduced, making it easier to record the precise state the mobile device currently is experiencing by giving access to all kinds of sensors.

If the goal is rather to investigate the theoretical overarching concepts of certain playback strategies or to scale up the test environment, one can utilize the simulative approach in the form of the LTE mobile streaming simulation framework. Initial measurements using this framework already suggest a high influence of the network QoS on the experienced streaming quality. A higher stalling probability caused by handovers during mobility was also uncovered.

7

CONCLUSIONS

Reliable streaming deviates from the established media streaming methods in a number of significant ways. In the past, streaming was dominated by server-controlled push-based approaches. Specialized and highly complex software and protocols — both proprietary, in the form of Flash and other players, and standards-based, using RTP — was needed to properly conduct streaming.

But many video streaming solutions deployed in the past few years have shifted to using simple client-side players better suited for the Web's ecosystem. Today, any Web browser can pull video data and display it directly inside a video element without the help of any additional software. The popularity of reliable video streaming services has increased in such a way that they are now one of the largest source of the Internet's traffic and are predicted to rise even further. But all these reliable streaming players — so-called due to them operating on top of the reliable TCP transport protocol — have in common that modeling, optimization, and measurement techniques used for unreliable streaming can not be applied to them any more due to their different approaches.

A second development in recent years is the increasing prevalence of cellular mobile networks, taking the form of UMTS and LTE. Smartphones have become omnipresent and traffic originating from these devices makes up an increasing portion of the Internet's total traffic. While the user plane traffic is still rather low when compared to wired Internet access, mobile networks have a much more intricate control plane and any user plane traffic causes a number of related control plane interactions which were discussed in 2.1. The actual volume of these signaling messages might be low, but they cause secondary load effects in the nodes that need to process the control plane data and store its state. And these effects are barely investigated or even factored into any network dimensioning and planning tasks.

When work on this dissertation thesis had started, both of these fields began to grow together. Smartphones developed to a state where watching video streams on their displays was not a hassle any more and there was sufficient capacity in the mobile access to transport video streams of relatively high quality in a timely fashion.

But again both also seemed to lack proper measurement and evaluation methods and models. This is what was tackled in this work.

One one side, light was shed on the control plane interactions of a typical 3G core network. The mobile networks' overly strong focus on signaling procedures and statefulness can be the source of numerous scalability issues as was previously explored for the radio interface. Here, through the analysis of network traces of a large operational mobile core network a novel definition of load in the core network was created. This load was put in relation to GTP tunnel management signaling between the two central core nodes SGSN and GGSN.

Diurnal variations and dependencies on the user's device and its user traffic characteristics could be observed in the trace. A queuing theory model that describes the load at a GGSN the load properties was derived from these findings and takes the form of a non-stationary Erlang loss model. With the help of this model a new spin on the GGSN's makeup using virtualization was introduced and tested in simulations to provide better load scaling options.

On the other side, a categorization approach and measurement model for reliable progressive video streaming that bases itself on emulating the streaming client's video playback and strategies, was introduced. With this at hand, a number of different strategies, modeled after existing real streaming solutions, were tested in a network emulation testbed and an interaction of the connection's latency and loss properties with the video streaming stalling characteristics was shown. This measurement model can be utilized to test new playback strategies for real applications.

Testing reliable video streaming in a mobile network and understanding its apparent behavior might be a very interesting combination of the two seemingly independent fields. But it also became clear during the evaluation of the mobile dataset that specific user traffic patterns can have a direct influence on the core's control plane. Segmented reliable streaming can show a certain ON-OFF style traffic pattern that may interact badly with the timers governing the control plane state and increase the core load.

Ultimately, this kind of negative interaction could not be completely verified or refuted during the course of this thesis due to the lack of data. However, valuable approaches on how to test for this were still shown in the form of active measurements enhanced with smartphone metadata and a streaming simulation framework on top of a ns-3-based LTE network.

Once issues, for example negative feedback loops between various components, have been discovered through these means, they can be mitigated and improved upon. For example, this can happen by adapting the playback strategies and protocols that are already being utilized — TCP is the best example of being constantly modified — by employing new protocols, such as DASH or HTTP/2, or utilizing cross-layer information to its fullest.

And these are also the lessons learned during the course of this thesis's work. Understanding and optimizing a certain item begins by investigating, evaluating and modeling existing iterations of it and mapping out the limits and possibilities. As reliable streaming in a mobile network is a relatively new application displaying very complex interactions and was generally not very well investigated, a lot of groundwork had to be done. And the measurement models provided here attempt to do just that. Future research can utilize the work conducted here as a foundation. It would be great to see, if the work spawns a number of scientific offspring that try to answer some of the questions and issues raised here. Even without further studies, a lasting benefit may already be present for both video streaming implementers and mobile network operators. They have now access to more tools to optimize their products for specific scenarios.

It was also shown that optimization in general does not automatically constitute making networks and applications more complex or putting more intelligence into the network. On the contrary, it is often better to uphold some core principles that have driven computer networking

and the Internet thus far. This includes "Keep it simple, stupid" (KISS), the focus on end-to-end systems, and also keeping a layered network stack.

Many of the issues with streaming in mobile networks discussed in this work have arisen by not adhering to these principles. The UMTS control plane is so tremendous, spanning a large number of dedicated network nodes and merging several protocol layers into one big specification that many undesirable interactions occur. The original developers could not have foreseen all future use cases and what side effects they might bring with them. On the flip side, this thesis would not even have been possible without the many peculiarities found in these 3G specifications.

7.1 Future Work

Alas, work on all these individual items discussed here is certainly not done. As is often the case in scientific work, progress will be made in small iterative steps. Therefore, this final section collects some planned work and ideas that follow up on the thesis's research angles in the future.

Concerning mobile core networks, most of the deductions were made on the basis of a single core network trace from one network and need to be verified with additional and more recent traces from other networks. This could simultaneously investigate the effects of any changes to the network in comparison to the original trace, for example regarding traffic and device composition or the influences of specific network parameters.

That trace did also lack some particular interesting details regarding the actual GTP IE data from the messages and control plane procedures on off-user-traffic-path links (e.g., to the HLR). Any subsequent passive measurement trace should therefore factor in more information and should provide measurements at several synchronized vantage points inside the network to better correlate the individual message. Additional probes at the radio or backhaul links could prove especially fruitful in this endeavor.

But as the usage UMTS is declining in favor of LTE, LTE-Advanced and soon even proposed 5G architectures, the core control plane in these networks also needs to be put under close scrutiny, and not just in a simulator as in this work. A multi-vantage point measurement campaign in an actual EPC can make an informed comparison to the existing 3G trace and find improvements or regressions. As the combination of SGSN and GGSN is replaced by SGW/PGW with a slightly different subset of functionalities a new core load queuing model should also be created in the course of the trace's statistical evaluation with these differences in mind. This can also lead to better load models once more core network elements are evaluated and factored into the queuing model — especially including the off-user-traffic path elements, including the HSS, MME, or PCRF.

Through this process, suggestions for future further iterations of the mobile network control plane specifications can be derived and taken into consideration. Any upcoming measurement campaign should also consider running a concurrent reliable video streaming active measure-

ment experiment in the network under study. This way, a correlation between streaming traffic and occurring control plane signaling could be drawn much more easily.

Moving on to the modeling and measurement of reliable streaming it is important to keep up with the current flavors of playback strategies. Now that adaptive streaming is being used more widespread adaptive strategies should also be taken into closer consideration. In addition, the changes on other protocol layers have to be constantly monitored for their potential influence on streaming. New protocols should be tested if they are viable for reliable streaming.

For example, HTTP/2's improved multiplexing features could be worthwhile for the simultaneous transport of multiple streaming segments. This is especially helpful when SVC is employed to provide the various quality levels in adaptive streaming. Multiplexing would allow multiple layers to be transported with differing priority simultaneously using just one connection.

The more strategies are known and understood the more candidates are available in the search for the best strategy fitting a given scenario or applicable to most cases. A good candidate here is probably an adaptive threshold strategy which observes its buffer fill rate in a sliding window.

When the base effects of the stalling characteristics are well understood, more complex adaptive streaming metrics could also be derived from this to better represent the QoE. A potential objective MOS computation could relate these basic statistics to subjective quality impressions gained from user studies as was already conducted in other publications for plain progressive streaming.

To better observe these interactions between streaming and mobile networks a lot of work still has to be done to bring mobile network simulators up to parity with actual networks. A better control plane representation in, for example, ns-3 or OMNeT++ would go a long way in increasing the validity of the results derived from these simulations. If a new, more fitting simulation platform arises, the reliable streaming model can also be reimplemented there as it is kept portable.

Similarly, simulators are also lacking behind in their representation of current higher layer protocols, especially TCP. Solutions need to be explored to remedy this and make the simulated stacks more comparable to the fast changing stacks of current OSs. Small changes here can have a noticeable impact on the application layer and can also falsify simulation results, as they are not applicable to any actual network.

Finally, regarding the ideas for a cross-layer information exchange, the general cross-layer information broker should be implemented on different architectures (e.g., an Android smartphone and a Linux laptop) to test different protocol stacks and different amounts of cross-layer information sources and sensory input. An actual implementation of the handover-aware adaptive playback strategy should also be relatively easy to achieve as a prototype to test the broker in a practical situation.

VITA

Florian Metzger received his Diplom degree in computer science at the University of Würzburg, Germany, in 2009. Since May 2010 he is research assistant at the Research Group of Future Communication at the University of Vienna, Austria. His doctoral thesis covers modelling mobile network signalling traffic and measuring TCP-based video streaming applications. His other research interests include sensor-aware network measurements and simulations as well as improvements to the mobile network protocol stacks. In the past he also worked on peer-to-peer location awareness as part of the work for his diploma thesis.

OWN PUBLICATIONS

Parts of the research conducted in this thesis is based on these already published publications.

[BFT12] A. Biernacki, **F. Metzger**, 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/.

[F M+11] **F. Metzger**, 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.

[F M+12] **F. Metzger**, 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: *University of Würzburg Institute of Computer Science Research Report Series* 484 (2012). URL: http://eprints.cs.univie.ac.at/3515/1/uniwue_tr_univie.pdf.

[F M+14] **F. Metzger**, A. Rafetseder, P. Romirer-Maierhofer, and K. Tutschku. "Exploratory Analysis of a GGSN's PDP Context Signaling Load". In: *Journal of Computer Networks and Communications* (Feb. 2014). DOI: doi:10.1155/2014/526231. URL: http://www.hindawi.com/journals/jcnc/2014/526231/.

[FRT12] **F. Metzger**, 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.

[FSH14] **F. Metzger**, C. Schwartz, and T. Hoßfeld. "GTP-based Load Model and Virtualization Gain for a Mobile Network's GGSN". In: *Communications and Electronics (ICCE), 2014 IEEE Fifth International Conference on.* July 2014, pp. 206–211. DOI: 10.1109/CCE.2014.6916704.

[Oec+09] S. Oechsner, F. Lehrieder, T. Hoßfeld, **F. Metzger**, D. Staehle, and K. Pussep. "Pushing the performance of Biased Neighbor Selection through Biased Unchoking". In: *P2P'09. IEEE Ninth International Conference on Peer-to-Peer Computing, 2009.* IEEE. Sept. 2009, pp. 301–310. DOI: 10.1109/P2P.2009.5284527.

[Raf+11] A. Rafetseder, **F. Metzger**, 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.* Cnet '11. San Francisco, California: International Teletraffic Congress, 2011,

pp. 31–36. ISBN: 978-0-9836283-1-6.
URL: http://dl.acm.org/citation.cfm?id=2043527.2043532.

[Raf+12] A. Rafetseder, **F. Metzger**, 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/.

[RFP13] A. Rafetseder, **F. Metzger**, and L. Pühringer. "Sensorium – A Generic Sensor Framework". In: *PIK - Praxis der Informationsverarbeitung und Kommunikation* 36.1 (Feb. 2013), p. 46.
DOI: doi:10.1515/pik-2012-0061.

SOURCE CODE

Thesis and all associated figure plotting scripts
Available at: https://github.com/fmetzger/thesis
License: Creative Commons Attribution-ShareAlike 4.0 International [CC BY-SA]

Video Streaming Emulation & Evaluation
Available at: https://github.com/fmetzger/videostreaming-bufferemulation
License: Unlicense http://unlicense.org

GGSN Queuing Simulation
Available at: https://github.com/fmetzger/ggsn-simulation
License: public domain

Sensorium
Available at: https://github.com/fmetzger/android-sensorium
License: GNU Lesser General Public License version 3 (LGPLv3) [LGPL]

Mobile Streaming Simulation
Available at: https://github.com/Steindi01/lteNS3
License: GNU General Public License (GPL) version 3 [GPL]

BIBLIOGRAPHY

[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.

[3GP11] 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), Dec. 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*. MMSys '11. San Jose, CA, USA: ACM, 2011, pp. 157–168. ISBN: 978-1-4503-0518-1. DOI: 10.1145/1943552.1943574. URL: http://doi.acm.org/10.1145/1943552.1943574.

[AF10] P. Arlos and M. Fiedler. "Influence of the Packet Size on the One-Way Delay in 3G Networks". In: *Passive and Active Measurement*. Ed. by A. Krishnamurthy and B. Plattner. Vol. 6032. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, pp. 61–70. ISBN: 978-3-642-12333-7. DOI: 10.1007/978-3-642-12334-4_7. URL: http://dx.doi.org/10.1007/978-3-642-12334-4_7.

[AJZ10] V. K. Adhikari, S. Jain, and Z.-L. Zhang. "YouTube Traffic Dynamics and Its Interplay with a Tier-1 ISP: An ISP Perspective". In: *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*. IMC '10. Melbourne, Australia: ACM, 2010, pp. 431–443. ISBN: 978-1-4503-0483-2. DOI: 10.1145/1879141.1879197. URL: http://doi.acm.org/10.1145/1879141.1879197.

[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.

[ASO14] S. Ahsan, V. Singh, and J. Ott.
"Characterizing Internet Video for Large-scale Active Measurements".
In: *ArXiv e-prints* (Aug. 2014). arXiv: 1408.5777 [cs.MM].

[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.

[Bal+13] N. Baldo, M. Requena-Esteso, M. Miozzo, and R. Kwan. "An Open Source Model for the Simulation of LTE Handover Scenarios and Algorithms in Ns-3".
In: *Proceedings of the 16th ACM International Conference on Modeling, Analysis & Simulation of Wireless and Mobile Systems*. MSWiM '13.
Barcelona, Spain: ACM, 2013, pp. 289–298. ISBN: 978-1-4503-2353-6.
DOI: 10.1145/2507924.2507940.
URL: http://doi.acm.org/10.1145/2507924.2507940.

[Bar64] P. Baran. "On Distributed Communications Networks".
In: *Communications Systems, IEEE Transactions on* 12.1 (Mar. 1964), pp. 1–9.
ISSN: 0096-1965. DOI: 10.1109/TCOM.1964.1088883.

[BC01] M. S. Blumenthal and D. D. Clark. "Rethinking the Design of the Internet: The End-to-end Arguments vs. The Brave New World".
In: *ACM Trans. Internet Technol.* 1.1 (Aug. 2001), pp. 70–109. ISSN: 1533-5399.
DOI: 10.1145/383034.383037.
URL: http://doi.acm.org/10.1145/383034.383037.

[BCZ97] S. Bhattacharjee, K. Calvert, and E. Zegura.
"Active networking and the end-to-end argument".
In: *Network Protocols, 1997. Proceedings., 1997 International Conference on*.
Oct. 1997, pp. 220–228. DOI: 10.1109/ICNP.1997.643717.

[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.

[BLH09] J.-M. Bonnin, I. Lassoued, and Z. B. Hamouda.
"Automatic Multi-interface Management Through Profile Handling".
In: *Mob. Netw. Appl.* 14.1 (Feb. 2009), pp. 4–17. ISSN: 1383-469X.
DOI: 10.1007/s11036-008-0117-6.
URL: http://dx.doi.org/10.1007/s11036-008-0117-6.

[BOP94] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson.
"TCP Vegas: New Techniques for Congestion Detection and Avoidance".
In: *SIGCOMM Comput. Commun. Rev.* 24.4 (Oct. 1994), pp. 24–35. ISSN: 0146-4833.
DOI: 10.1145/190809.190317.
URL: http://doi.acm.org/10.1145/190809.190317.

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

[BP13] M. Belshe and R. Peon. *SPDY Protocol — Draft 3.1*. 2013. URL: http://www.chromium.org/spdy/spdy-protocol/spdy-protocol-draft3-1.

[BPT14] M. Belshe, R. Peon, and M. Thomson. *Hypertext Transfer Protocol version 2.0*.
Internet-Draft. Internet Engineering Task Force, Apr. 2014.
URL: http://tools.ietf.org/html/draft-ietf-httpbis-http2.

[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.

[Cap+14] J. Cappos, L. Wang, R. Weiss, Y. Yang, and Y. Zhuang.
"BlurSense: Dynamic fine-grained access control for smartphone privacy".
In: *Sensors Applications Symposium (SAS), 2014 IEEE*. Feb. 2014, pp. 329–332.
DOI: 10.1109/SAS.2014.6798970.

[Cap09] Cappos, J. and Beschastnikh, I. and Krishnamurthy, A. and Anderson, T.
"Seattle: a platform for educational cloud computing". In: *Proceedings of the 40th ACM technical symposium on Computer science education*. SIGCSE '09.
Chattanooga, TN, USA: ACM, 2009, pp. 111–115. ISBN: 978-1-60558-183-5.
URL: http://doi.acm.org/10.1145/1508865.1508905.

[CC BY-SA] *Attribution-ShareAlike International*. Version 4.0. Creative Commons.
URL: https://creativecommons.org/licenses/by-sa/4.0/.

[Che+14] Y. Cheng, J. Chu, S. Radhakrishnan, and A. Jain. *TCP Fast Open*. Internet-Draft. Internet Engineering Task Force, Mar. 2014.
URL: http://tools.ietf.org/html/draft-ietf-tcpm-fastopen.

[Chu+03] B. Chun et al. "PlanetLab: An Overlay Testbed for Broad-coverage Services". In: *SIGCOMM Comput. Commun. Rev.* 33.3 (July 2003), pp. 3–12. ISSN: 0146-4833.
DOI: 10.1145/956993.956995.
URL: http://doi.acm.org/10.1145/956993.956995.

[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.

[Cis14] Cisco. *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2013–2018*. Feb. 2014.
URL: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.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.

[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.

[DMW95] J. L. Davis, W. A. Massey, and W. Whitt. "Sensitivity to the Service-Time Distribution in the Nonstationary Erlang Loss Model". In: *Management Science* 41.6 (1995), pp. 1107–1116. ISSN: 00251909.
URL: http://www.jstor.org/stable/2632837.

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

[Enc+14] W. Enck et al. "TaintDroid: An Information Flow Tracking System for Real-time Privacy Monitoring on Smartphones". In: vol. 57. 3.

New York, NY, USA: ACM, Mar. 2014, pp. 99–106. DOI: 10.1145/2494522.
URL: http://doi.acm.org/10.1145/2494522.

[Erl17] A. K. Erlang. "Solution of some problems in the theory of probabilities of significance in automatic telephone exchanges".
In: *Elektrotkeknikeren* 13 (1917), pp. 5–13.

[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.

[Fel10] A. Feldmann. "The Internet Architecture - Is a Redesign Needed?" German.
In: *Netzwelt - Wege, Werte, Wandel*.
Ed. by D. Klumpp, H. Kubicek, A. Roßnagel, and W. Schulz.
Springer Berlin Heidelberg, 2010, pp. 147–160. ISBN: 978-3-642-05053-4.
DOI: 10.1007/978-3-642-05054-1_11.
URL: http://dx.doi.org/10.1007/978-3-642-05054-1_11.

[FHT10] M. Fiedler, T. Hoßfeld, and P. Tran-Gia. "A generic quantitative relationship between quality of experience and quality of service".
In: *Network, IEEE* 24.2 (Mar. 2010), pp. 36–41. ISSN: 0890-8044.
DOI: 10.1109/MNET.2010.5430142.

[FMF12] A. Field, J. Miles, and Z. Field. *Discovering statistics using R*.
Sage publications, 2012.

[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. DOI: 10.1109/ICIP.2008.4711777.

[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*.
IMC '07. San Diego, California, USA: ACM, 2007, pp. 15–28.
ISBN: 978-1-59593-908-1. DOI: 10.1145/1298306.1298310.
URL: http://doi.acm.org/10.1145/1298306.1298310.

[GK11] D. Groenewegen and H. Kleppe.
Detecting and quantifying bufferbloat in network paths. 2011.

[GN11] J. Gettys and K. Nichols. "Bufferbloat: Dark Buffers in the Internet".
In: *Queue* 9.11 (Nov. 2011), 40:40–40:54. ISSN: 1542-7730.
DOI: 10.1145/2063166.2071893.
URL: http://doi.acm.org/10.1145/2063166.2071893.

[GPL] *GNU General Public License*. Version 3. Free Software Foundation, June 29, 2007.
URL: https://www.gnu.org/licenses/gpl.html.

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

[Har+06] L. Haratcherev, J. Taal, K. Langendoen, R. Lagendijk, and H. Sips. "Optimized video streaming over 802.11 by cross-layer signaling". In: *Communications Magazine, IEEE* 44.1 (2006), pp. 115–121. ISSN: 0163-6804. DOI: 10.1109/MCOM.2006.1580941.

[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: *NETWORKING 2012*. Ed. by R. Bestak, L. Kencl, L. Li, J. Widmer, and H. Yin. Vol. 7289. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, pp. 318–330. ISBN: 978-3-642-30044-8. DOI: 10.1007/978-3-642-30045-5_24. URL: http://dx.doi.org/10.1007/978-3-642-30045-5_24.

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

[HHM10] K. A. Hummel, A. Hess, and H. Meyer. "Mobilität im „Future Internet"". German. In: *Informatik-Spektrum* 33.2 (2010), pp. 143–159. ISSN: 0170-6012. DOI: 10.1007/s00287-010-0425-7. URL: http://dx.doi.org/10.1007/s00287-010-0425-7.

[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/.

[Hoß+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.

[Hoß+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.

[Hoß+13] T. Hoßfeld, R. Schatz, E. Biersack, and L. Plissonneau. "Internet Video Delivery in YouTube: From Traffic Measurements to Quality of Experience". English. In: *Data Traffic Monitoring and Analysis*. Ed. by E. Biersack, C. Callegari, and M. Matijasevic. Vol. 7754. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, pp. 264–301. ISBN: 978-3-642-36783-0. DOI: 10.1007/978-3-642-36784-7_11. URL: http://dx.doi.org/10.1007/978-3-642-36784-7_11.

[HR11] S. Ha and I. Rhee. “Taming the elephants: New TCP slow start”. In: *Computer Networks* 55.9 (2011), pp. 2092–2110. ISSN: 1389-1286. DOI: http://dx.doi.org/10.1016/j.comnet.2011.01.014. URL: http://www.sciencedirect.com/science/article/pii/S1389128611000363.

[HRX08] S. Ha, I. Rhee, and L. Xu. “CUBIC: A New TCP-friendly High-speed TCP Variant”. In: *SIGOPS Oper. Syst. Rev.* 42.5 (July 2008), pp. 64–74. ISSN: 0163-5980. DOI: 10.1145/1400097.1400105. URL: http://doi.acm.org/10.1145/1400097.1400105.

[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.

[HW05] E. Halepovic and C. Williamson. “Characterizing and Modeling User Mobility in a Cellular Data Network”. In: *Proceedings of the 2Nd ACM International Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks*. PE-WASUN ’05. Montreal, Quebec, Canada: ACM, 2005, pp. 71–78. ISBN: 1-59593-182-1. DOI: 10.1145/1089803.1089969. URL: http://doi.acm.org/10.1145/1089803.1089969.

[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.

[ITU13] ITU. *ITU-T Recommendation P.1202.1: Parametric non-intrusive bitstream assessment of video media streaming quality - Lower resolution application area.* Tech. rep. International Telecommunication Union, 2013.
URL: http://www.itu.int/rec/T-REC-P.1202.1-201210-I/en.

[Jac88] V. Jacobson. "Congestion Avoidance and Control".
In: *SIGCOMM Comput. Commun. Rev.* 18.4 (Aug. 1988), pp. 314–329.
ISSN: 0146-4833. DOI: 10.1145/52325.52356.
URL: http://doi.acm.org/10.1145/52325.52356.

[Jar+11] M. Jarschel, D. Schlosser, S. Scheuring, and T. Hoßfeld.
"An Evaluation of QoE in Cloud Gaming Based on Subjective Tests".
In: *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2011 Fifth International Conference on.* June 2011, pp. 330–335.
DOI: 10.1109/IMIS.2011.92.

[Jeo+12] J. Jeon et al.
"Dr. Android and Mr. Hide: Fine-grained Permissions in Android Applications".
In: *Proceedings of the Second ACM Workshop on Security and Privacy in Smartphones and Mobile Devices.* SPSM '12.
Raleigh, North Carolina, USA: ACM, 2012, pp. 3–14. ISBN: 978-1-4503-1666-8.
DOI: 10.1145/2381934.2381938.
URL: http://doi.acm.org/10.1145/2381934.2381938.

[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.

[KHF06] E. Kohler, M. Handley, and S. Floyd.
"Designing DCCP: Congestion Control Without Reliability".
In: *SIGCOMM Comput. Commun. Rev.* 36.4 (Aug. 2006), pp. 27–38.
ISSN: 0146-4833. DOI: 10.1145/1151659.1159918.
URL: http://doi.acm.org/10.1145/1151659.1159918.

[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.

[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). ISSN: 0146-4833. URL: http://dl.acm.org/citation.cfm?id=2043164.1851194.

[Lan+11] M. Laner, P. Svoboda, E. Hasenleithner, and M. Rupp. "Dissecting 3G Uplink Delay by Measuring in an Operational HSPA Network". In: *Passive and Active Measurement*. Ed. by N. Spring and G. Riley. Vol. 6579. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2011, pp. 52–61. ISBN: 978-3-642-19259-3. DOI: 10.1007/978-3-642-19260-9_6. URL: http://dx.doi.org/10.1007/978-3-642-19260-9_6.

[Lan+12a] 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.

[Lan+12b] M. Laner, P. Svoboda, S. Schwarz, and M. Rupp. "Users in cells: A data traffic analysis". In: *Wireless Communications and Networking Conference (WCNC), 2012 IEEE*. Apr. 2012, pp. 3063–3068. DOI: 10.1109/WCNC.2012.6214330.

[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*. May 2007, pp. 1289–1297. DOI: 10.1109/INFCOM.2007.153.

[LGPL] *GNU Lesser General Public License*. Version 3.
Free Software Foundation, June 29, 2007.
URL: https://www.gnu.org/licenses/lgpl.html.

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

[LK05] J. Landman and P. Kritzinger.
"Delay analysis of downlink {IP} traffic on {UMTS} mobile networks".
In: *Performance Evaluation* 62.1–4 (2005). Performance 2005 24th International Symposium on Computer Performance, Modeling, Measurements and Evaluation, pp. 68–82. ISSN: 0166-5316.
DOI: http://dx.doi.org/10.1016/j.peva.2005.07.007. URL: http://www.sciencedirect.com/science/article/pii/S0166531605000866.

[LL00] 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.

[Meh+11] C. Mehlführer, J. Colom Colom Ikuno, M. Šimko, S. Schwarz, M. Wrulich, and M. Rupp. "The Vienna LTE simulators - Enabling reproducibility in wireless communications research".
In: *EURASIP Journal on Advances in Signal Processing* 2011.1, 29 (2011).
DOI: 10.1186/1687-6180-2011-29.
URL: http://dx.doi.org/10.1186/1687-6180-2011-29.

[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". English. In: *Traffic Monitoring and Analysis*. Ed. by F. Ricciato, M. Mellia, and E. Biersack. Vol. 6003. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, pp. 17–31. ISBN: 978-3-642-12364-1. DOI: 10.1007/978-3-642-12365-8_2. URL: http://dx.doi.org/10.1007/978-3-642-12365-8_2.

[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.

[Nan+13] A. Nandugudi et al. "PhoneLab: A Large Programmable Smartphone Testbed". In: *Proceedings of First International Workshop on Sensing and Big Data Mining*. SENSEMINE'13. Roma, Italy: ACM, 2013, 4:1–4:6. ISBN: 978-1-4503-2430-4. DOI: 10.1145/2536714.2536718. URL: http://doi.acm.org/10.1145/2536714.2536718.

[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.

[NJ14] K. Nichols and V. Jacobson. *Controlled Delay Active Queue Management*. Internet-Draft. Internet Engineering Task Force, Mar. 2014. URL: http://tools.ietf.org/html/draft-nichols-tsvwg-codel.

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

[Pau+11] U. Paul, A. Subramanian, M. Buddhikot, and S. Das. "Understanding traffic dynamics in cellular data networks". In: *INFOCOM, 2011 Proceedings IEEE*. Apr. 2011, pp. 882–890. DOI: 10.1109/INFCOM.2011.5935313.

[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.

[PPF05] D. Pesch, M. I. Pous, and G. Foster.
"Performance evaluation of SIP-based multimedia services in {UMTS}".
In: *Computer Networks* 49.3 (2005). Selected Papers from the European Wireless 2004 Conference European Wireless 2004, pp. 385–403. ISSN: 1389-1286.
DOI: http://dx.doi.org/10.1016/j.comnet.2005.05.013. URL: http://www.sciencedirect.com/science/article/pii/S1389128605001313.

[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*. MobiSys '11. Bethesda, Maryland, USA: ACM, 2011, pp. 321–334.
ISBN: 978-1-4503-0643-0. DOI: 10.1145/1999995.2000026.
URL: http://doi.acm.org/10.1145/1999995.2000026.

[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.

[RFC1191] J. Mogul and S. Deering. *Path MTU discovery*. RFC 1191 (Draft Standard).
Internet Engineering Task Force, Nov. 1990.
URL: http://www.ietf.org/rfc/rfc1191.txt.

[RFC1889] 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.

[RFC2326] 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.

[RFC2327] 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.

[RFC2616] R. Fielding et al. *Hypertext Transfer Protocol – HTTP/1.1*.
RFC 2616 (Draft Standard). Obsoleted by RFCs 7230, 7231, 7232, 7233, 7234, 7235, updated by RFCs 2817, 5785, 6266, 6585.
Internet Engineering Task Force, June 1999.
URL: http://www.ietf.org/rfc/rfc2616.txt.

[RFC2974] 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.

[RFC3168] K. Ramakrishnan, S. Floyd, and D. Black. *The Addition of Explicit Congestion Notification (ECN) to IP*.
RFC 3168 (Proposed Standard). Updated by RFCs 4301, 6040.
Internet Engineering Task Force, Sept. 2001.
URL: http://www.ietf.org/rfc/rfc3168.txt.

[RFC3261] 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.

[RFC3393] 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.

[RFC3550] 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, 7160, 7164.
Internet Engineering Task Force, July 2003.
URL: http://www.ietf.org/rfc/rfc3550.txt.

[RFC3640] 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.

[RFC3711] M. Baugher, D. McGrew, M. Naslund, E. Carrara, and K. Norrman.
The Secure Real-time Transport Protocol (SRTP). RFC 3711 (Proposed Standard).
Updated by RFCs 5506, 6904. Internet Engineering Task Force, Mar. 2004.
URL: http://www.ietf.org/rfc/rfc3711.txt.

[RFC4033] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose.
DNS Security Introduction and Requirements. RFC 4033 (Proposed Standard).
Updated by RFCs 6014, 6840. Internet Engineering Task Force, Mar. 2005.
URL: http://www.ietf.org/rfc/rfc4033.txt.

[RFC4340] 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.

[RFC4445] 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.

[RFC4604] 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.

[RFC5245] 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.

[RFC5246] T. Dierks and E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.2*. RFC 5246 (Proposed Standard). Updated by RFCs 5746, 5878, 6176. Internet Engineering Task Force, Aug. 2008.
URL: http://www.ietf.org/rfc/rfc5246.txt.

[RFC5389] J. Rosenberg, R. Mahy, P. Matthews, and D. Wing. *Session Traversal Utilities for NAT (STUN)*. RFC 5389 (Proposed Standard). Updated by RFC 7350. Internet Engineering Task Force, Oct. 2008.
URL: http://www.ietf.org/rfc/rfc5389.txt.

[RFC5482] L. Eggert and F. Gont. *TCP User Timeout Option*. RFC 5482 (Proposed Standard). Internet Engineering Task Force, Mar. 2009.
URL: http://www.ietf.org/rfc/rfc5482.txt.

[RFC5681] M. Allman, V. Paxson, and E. Blanton. *TCP Congestion Control*. RFC 5681 (Draft Standard). Internet Engineering Task Force, Sept. 2009.
URL: http://www.ietf.org/rfc/rfc5681.txt.

[RFC5682] P. Sarolahti, M. Kojo, K. Yamamoto, and M. Hata. *Forward RTO-Recovery (F-RTO): An Algorithm for Detecting Spurious Retransmission Timeouts with TCP*. RFC 5682 (Proposed Standard). Internet Engineering Task Force, Sept. 2009.
URL: http://www.ietf.org/rfc/rfc5682.txt.

[RFC5827] M. Allman, K. Avrachenkov, U. Ayesta, J. Blanton, and P. Hurtig. *Early Retransmit for TCP and Stream Control Transmission Protocol (SCTP)*. RFC 5827 (Experimental). Internet Engineering Task Force, May 2010.
URL: http://www.ietf.org/rfc/rfc5827.txt.

[RFC6120] 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.

[RFC6121] 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.

[RFC6184] 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.

[RFC6189] P. Zimmermann, A. Johnston, and J. Callas. *ZRTP: Media Path Key Agreement for Unicast Secure RTP*. RFC 6189 (Informational). Internet Engineering Task Force, Apr. 2011.
URL: http://www.ietf.org/rfc/rfc6189.txt.

[RFC6298] V. Paxson, M. Allman, J. Chu, and M. Sargent. *Computing TCP's Retransmission Timer*. RFC 6298 (Proposed Standard).

Internet Engineering Task Force, June 2011.
URL: http://www.ietf.org/rfc/rfc6298.txt.

[RFC6347] E. Rescorla and N. Modadugu. *Datagram Transport Layer Security Version 1.2*. RFC 6347 (Proposed Standard). Internet Engineering Task Force, Jan. 2012.
URL: http://www.ietf.org/rfc/rfc6347.txt.

[RFC6416] 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.

[RFC6455] I. Fette and A. Melnikov. *The WebSocket Protocol*. RFC 6455 (Proposed Standard). Internet Engineering Task Force, Dec. 2011.
URL: http://www.ietf.org/rfc/rfc6455.txt.

[RFC6582] T. Henderson, S. Floyd, A. Gurtov, and Y. Nishida. *The NewReno Modification to TCP's Fast Recovery Algorithm*. RFC 6582 (Proposed Standard). Internet Engineering Task Force, Apr. 2012.
URL: http://www.ietf.org/rfc/rfc6582.txt.

[RFC6698] P. Hoffman and J. Schlyter. *The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA*. RFC 6698 (Proposed Standard). Updated by RFC 7218. Internet Engineering Task Force, Aug. 2012.
URL: http://www.ietf.org/rfc/rfc6698.txt.

[RFC6733] V. Fajardo, J. Arkko, J. Loughney, and G. Zorn. *Diameter Base Protocol*. RFC 6733 (Proposed Standard). Updated by RFC 7075. Internet Engineering Task Force, Oct. 2012.
URL: http://www.ietf.org/rfc/rfc6733.txt.

[RFC6817] 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.

[RFC6824] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure. *TCP Extensions for Multipath Operation with Multiple Addresses*. RFC 6824 (Experimental). Internet Engineering Task Force, Jan. 2013.
URL: http://www.ietf.org/rfc/rfc6824.txt.

[RFC6830] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis. *The Locator/ID Separation Protocol (LISP)*. RFC 6830 (Experimental). Internet Engineering Task Force, Jan. 2013.
URL: http://www.ietf.org/rfc/rfc6830.txt.

[RFC6937] M. Mathis, N. Dukkipati, and Y. Cheng. *Proportional Rate Reduction for TCP*. RFC 6937 (Experimental). Internet Engineering Task Force, May 2013.
URL: http://www.ietf.org/rfc/rfc6937.txt.

[RFC6973] A. Cooper et al. *Privacy Considerations for Internet Protocols.* RFC 6973 (Informational). Internet Engineering Task Force, July 2013. URL: http://www.ietf.org/rfc/rfc6973.txt.

[RFC6982] J. Chu, N. Dukkipati, Y. Cheng, and M. Mathis. *Increasing TCP's Initial Window.* RFC 6928 (Experimental). Internet Engineering Task Force, Apr. 2013. URL: http://www.ietf.org/rfc/rfc6928.txt.

[RFC7021] C. Donley, L. Howard, V. Kuarsingh, J. Berg, and J. Doshi. *Assessing the Impact of Carrier-Grade NAT on Network Applications.* RFC 7021 (Informational). Internet Engineering Task Force, Sept. 2013. URL: http://www.ietf.org/rfc/rfc7021.txt.

[RFC7230] R. Fielding and J. Reschke. *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing.* RFC 7230 (Proposed Standard). Internet Engineering Task Force, June 2014. URL: http://www.ietf.org/rfc/rfc7230.txt.

[RFC7231] R. Fielding and J. Reschke. *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content.* RFC 7231 (Proposed Standard). Internet Engineering Task Force, June 2014. URL: http://www.ietf.org/rfc/rfc7231.txt.

[RFC7232] R. Fielding and J. Reschke. *Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests.* RFC 7232 (Proposed Standard). Internet Engineering Task Force, June 2014. URL: http://www.ietf.org/rfc/rfc7232.txt.

[RFC7233] R. Fielding, Y. Lafon, and J. Reschke. *Hypertext Transfer Protocol (HTTP/1.1): Range Requests.* RFC 7233 (Proposed Standard). Internet Engineering Task Force, June 2014. URL: http://www.ietf.org/rfc/rfc7233.txt.

[RFC7234] R. Fielding, M. Nottingham, and J. Reschke. *Hypertext Transfer Protocol (HTTP/1.1): Caching.* RFC 7234 (Proposed Standard). Internet Engineering Task Force, June 2014. URL: http://www.ietf.org/rfc/rfc7234.txt.

[RFC7235] R. Fielding and J. Reschke. *Hypertext Transfer Protocol (HTTP/1.1): Authentication.* RFC 7235 (Proposed Standard). Internet Engineering Task Force, June 2014. URL: http://www.ietf.org/rfc/rfc7235.txt.

[RFC7236] J. Reschke. *Initial Hypertext Transfer Protocol (HTTP) Authentication Scheme Registrations.* RFC 7236 (Informational). Internet Engineering Task Force, June 2014. URL: http://www.ietf.org/rfc/rfc7236.txt.

[RFC7237] J. Reschke. *Initial Hypertext Transfer Protocol (HTTP) Method Registrations.* RFC 7237 (Informational). Internet Engineering Task Force, June 2014. URL: http://www.ietf.org/rfc/rfc7237.txt.

[RFC7238] J. Reschke. *The Hypertext Transfer Protocol Status Code 308 (Permanent Redirect).* RFC 7238 (Experimental). Internet Engineering Task Force, June 2014. URL: http://www.ietf.org/rfc/rfc7238.txt.

[RFC7239] A. Petersson and M. Nilsson. *Forwarded HTTP Extension.* RFC 7239 (Proposed Standard). Internet Engineering Task Force, June 2014. URL: http://www.ietf.org/rfc/rfc7239.txt.

[RFC793] J. Postel. *Transmission Control Protocol.* RFC 793 (INTERNET STANDARD). Updated by RFCs 1122, 3168, 6093, 6528. Internet Engineering Task Force, Sept. 1981. URL: http://www.ietf.org/rfc/rfc793.txt.

[RFC896] J. Nagle. *Congestion Control in IP/TCP Internetworks.* RFC 896. Internet Engineering Task Force, Jan. 1984. URL: http://www.ietf.org/rfc/rfc896.txt.

[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+06] F. Ricciato et al. "Traffic monitoring and analysis in 3G networks: lessons learned from the METAWIN project". In: *e & i Elektrotechnik und Informationstechnik* 123.7-8 (2006), pp. 288–296. ISSN: 0932-383X. DOI: 10.1007/s00502-006-0362-y. URL: http://dx.doi.org/10.1007/s00502-006-0362-y.

[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.

[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.

[San14] Sandvine. *Sandvine Global Internet Phenomena Reports*. 2011–2014. URL: https://www.sandvine.com/trends/global-internet-phenomena/.

[SBP13] M. Seyedebrahimi, C. Bailey, and X.-H. Peng. "Model and Performance of a No-Reference Quality Assessment Metric for Video Streaming". In: *Circuits and Systems for Video Technology, IEEE Transactions on* 23.12 (Dec. 2013), pp. 2034–2043. ISSN: 1051-8215. DOI: 10.1109/TCSVT.2013.2270365.

[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* 55.8 (2011), pp. 1921–1940. ISSN: 1389-1286. DOI: http://dx.doi.org/10.1016/j.comnet.2011.02.002. URL: http://www.sciencedirect.com/science/article/pii/S1389128611000491.

[Seu+13] M. Seufert, M. Slanina, S. Egger, and M. Kottkamp. ""To pool or not to pool": A comparison of temporal pooling methods for HTTP adaptive video streaming". In: *Quality of Multimedia Experience (QoMEX), 2013 Fifth International Workshop on*. July 2013, pp. 52–57. DOI: 10.1109/QoMEX.2013.6603210.

[Sha+11] M. Z. Shafiq, L. Ji, A. X. Liu, and J. Wang. "Characterizing and Modeling Internet Traffic Dynamics of Cellular Devices". In: *Proceedings of the ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems*. SIGMETRICS '11. San Jose, California, USA: ACM, 2011, pp. 305–316. ISBN: 978-1-4503-0814-4. DOI: 10.1145/1993744.1993776. URL: http://doi.acm.org/10.1145/1993744.1993776.

[Sha+12] 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.

[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. H. Saltzer, D. P. Reed, and D. D. Clark.
“End-to-end Arguments in System Design”.
In: *ACM Trans. Comput. Syst.* 2.4 (Nov. 1984), pp. 277–288. ISSN: 0734-2071.
DOI: 10.1145/357401.357402.
URL: http://doi.acm.org/10.1145/357401.357402.

[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.

[Str+13] A. Striegel, S. Liu, L. Meng, C. Poellabauer, D. Hachen, and O. Lizardo.
“Lessons Learned from the Netsense Smartphone Study”.
In: *Proceedings of the 5th ACM Workshop on HotPlanet*. HotPlanet ’13.
Hong Kong, China: ACM, 2013, pp. 51–56. ISBN: 978-1-4503-2177-8.
DOI: 10.1145/2491159.2491171.
URL: http://doi.acm.org/10.1145/2491159.2491171.

[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.

[SZS06] K. T. J. Song, Q. Zhang, and M. Sridharan. "Compound TCP: A scalable and TCP-friendly congestion control for high-speed networks". In: *Proceedings of PFLDnet 2006* (2006).

[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.

[Tra05] P. Tran-Gia. *Einführung in die Leistungsbewertung und Verkehrstheorie*. Oldenbourg Verlag, 2005.

[TT05] J. S. Turner and D. E. Taylor. "Diversifying the internet". In: *Global Telecommunications Conference, 2005. GLOBECOM'05. IEEE*. Vol. 2. IEEE. 2005, 6–pp. DOI: 10.1109/GLOCOM.2005.1577741.

[Vos00] D. Vose. *Risk analysis: A quantitative guide*. 2000.

[VŠR11] M. Vranješ, T. Švedek, and S. Rimac-Drlje. "The use of NS-2 simulator in studying UMTS performances". In: *International Journal of Electrical and Computer Engineering Systems* 1.2 (2011), pp. 63–71.

[W3C13] W3C. *WebRTC 1.0: Real-time Communication Between Browsers*. Working Draft. Sept. 2013. URL: http://www.w3.org/TR/webrtc/.

[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+11] Z. Wang, Z. Qian, Q. Xu, Z. Mao, and M. Zhang. "An Untold Story of Middleboxes in Cellular Networks". In: *SIGCOMM Comput. Commun. Rev.* 41.4 (Aug. 2011), pp. 374–385. ISSN: 0146-4833. DOI: 10.1145/2043164.2018479. URL: http://doi.acm.org/10.1145/2043164.2018479.

[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*. Nov. 2009, pp. 1–7. DOI: 10.1109/GLOCOM.2009.5425784.

[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.

[Xu+11] Q. Xu, J. Huang, Z. Wang, F. Qian, A. Gerber, and Z. M. Mao. “Cellular Data Network Infrastructure Characterization and Implication on Mobile Content Placement”. In: *Proceedings of the ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems*. SIGMETRICS ’11. San Jose, California, USA: ACM, 2011, pp. 317–328. ISBN: 978-1-4503-0814-4. DOI: 10.1145/1993744.1993777. URL: http://doi.acm.org/10.1145/1993744.1993777.

[Yan11] C. Yang. *Weather the signaling storm*. 2011. URL: http://www.huawei.com/en/static/hw-094153.pdf.

[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.