TECHNISCHE
UNIVERSITÄT
WIEN
Vienna University of Technology

Doctoral Thesis

Analyzing Packet Delay in Reactive Networks

Author
Markus Laner
0325687

Lammweg 28/B
39050 Girlan, IT
∗ Bozen, 7.6.1984

A thesis submitted in fulfillment of the requirements
for the degree of Doctor of Philosophy

to the

Faculty of Electrical Engineering and Information Technology
Vienna University of Technology

Vienna, September 2013

Dissertation

Analyzing Packet Delay in Reactive Networks

Autor
Markus Laner
0325687

Lammweg 28/B
39050 Girlan, IT
∗ Bozen, 7.6.1984

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Doktors der technischen Wissenschaften

eingereicht an der

Fakultät für Elektrotechnik und Informationstechnik
Technischen Universität Wien

Wien, September 2013

Advisor **Markus Rupp**

Institute of Telecommunications
Vienna University of Technology, Austria

Examiner **Raymond Knopp**

Mobile Communications Department
EURECOM, France

Declaration of Authorship

I, Markus Laner, declare that this thesis entitled, "Analyzing Packet Delay in Reactive Networks" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

Vienna University of Technology

Abstract

Faculty of Electrical Engineering and Information Technology
Institute of Telecommunications

Analyzing Packet Delay
in Reactive Networks

by Markus Laner

Packet delay is among the most important characteristics of networks, since it directly impacts the user satisfaction. Accordingly, it is often used for benchmarking networks.

Modern mobile cellular networks handle data streams with complex algorithms, in order to maximize the achievable throughput. Considering that, two question arise: (i) How can latency be measured in such networks? (ii) Are respective concepts from wired networks directly applicable to wireless networks?

This thesis provides a self-contained guide on how to measure packet delay in mobile cellular networks. Covered aspects reach from timekeeping on tracing hardware to the design of measurement processes.

Measurements in operational networks have evidenced them to be reactive; namely, the experienced latency depends on the injected traffic pattern (including the respective history). Accordingly, involved actions are required to obtain fair delay benchmarks, as outlined in this thesis. Thereby, the design of traffic patterns is regarded as integral part of the measurement methodology.

I conclude that several well-known approaches for delay measurements are not sufficient for reactive networks and may produce misleading results. The presented concepts contribute to the topic of latency measurement methodologies, such that they keep pace with the fast evolution of communication technologies.

Technischen Universität Wien

Kurzfassung

Fakultät für Elektrotechnik und Informationstechnik
Institute of Telecommunications

Analyzing Packet Delay
in Reactive Networks

von Markus Laner

Verzögerungszeiten gehören zu den wichtigsten Kenngrößen von Netzwerken für Paketübertragung. Sie haben direkten Einfluss auf die Zufriedenheit der Kunden und werden deshalb häufig für Vergleichstests (Benchmarks) herangezogen.

Hochmoderne mobile Netzwerke verwenden komplexe Algorithmen um die erreichbaren Datenraten zu maximieren. In diesen Fällen stellen sich zwei Fragen: (i) Wie können Verzögerungszeiten sinnvoll gemessen werden? (ii) Gelten die entsprechenden Konzepte, die für kabelgebundene Netzwerke entwickelt wurden auch in diesen Fällen?

Diese Dissertationsschrift gibt einen geschlossenen Überblick über die Messung von Verzögerungszeiten in mobilen kabellosen Netzwerken. Die behandelten Themen reichen von präziser Zeiterfassung bis zum Design von passenden Messmethoden.

Messungen in öffentlichen Netzwerken haben gezeigt, dass sich die erwähnten Netze reaktiv verhalten. Das heißt, die Verzögerungszeiten welche einzelne Pakete erfahren hängen maßgeblich vom Verkehrsmuster im Datenstrom ab. Im Besonderen haben die vorherigen Pakete starken Einfluss auf die Verzögerungszeit des aktuellen Pakets. Dieser Sachverhalt erschwert die Durchführung fairer Vergleichstests. Die vorliegende Arbeit diskutiert solche Vergleichstests und erläutert entsprechende Messmethoden. Dabei ist das Design von Verkehrsmustern ein zentraler Bestandteil.

Bekannte Messmethoden reichen nicht aus, um Verzögerungszeiten in mobilen Netzwerken tadellos zu erfassen. Sie führen zu unvollständigen Ergebnissen. Die hier vorgestellten Messmethoden beheben diese Defizite. Sie leisten ihren Beitrag dazu, dass Methoden zur Messung von Verzögerungszeiten mit der rasanten Entwicklung von Kommunikationstechnologien schritthalten können.

Acknowledgements

This thesis would not have been possible without the support of some great people, which I would like to thank at this place.

First of all, thanks to all my colleagues who helped me to establish the scientific content of this thesis, namely: Philipp Svoboda, Markus Rupp, Navid Nikaein, Fabio Ricciato, Joachim Fabini, Peter Romirer-Maierhofer, Eduard Hasenleithner, Stefan Schwarz and Sebastian Caban. Although the following pages are written in the singular form, considerable contributions are based on their inputs. Especially, I am in debt to Philipp Svoboda and Markus Rupp, who helped me with innumerable fruitful discussions and encouraged me in the difficult phases of my PhD studies. Further, I convey my thanks to Markus Rupp and Raymond Knopp for the evaluation of this thesis.

The present work is based on experiences obtained from several projects, namely, the *LOLA*[1] project, funded by the EU–FP7 framework, as well as, the *DARWIN++*, *DARWIN 3* and *DARWIN 4* projects[2], funded by the FFG–Comet framework. I owe special thanks to the people behind these projects, who organized fundings, wrote proposals and spent their time for creating an outstanding working environment. Those are: Philipp Svoboda, Markus Rupp, Navid Nikeain, Raymond Knopp, Fabio Ricciato and Waltraud Müllner.

My colleagues at the *Institute of Telecommunications* created a harmonic environment, reaching from scientific discussions to leisure activities, for which I would like to thank them. Finally, and most importantly, I want to thank my family as well as all my friends, for their continuous support.

[1] www.ict-lola.eu
[2] www.ftw.at

Contents

Chapter 1

Introduction

Delay is one of the most important characteristics of any kind of connection on which information is exchanged, since it directly influences user satisfaction. Modern packet switched networks, as considered in this thesis, require one-way delays in the sub-millisecond range to satisfy the demands of real-time applications. At present, the most popular real-time applications are *telephony*, *online gaming* and *video streaming*; in the future, several new applications will appear, for example *machine-to-machine communications*.
The increased use of mobile devices entails that many Internet users rely on cellular mobile networks. With the advent of the 4th generation of mobile networks, such as 3GPP Long Term Evolution (LTE), the network designers pay increased attention to the reduction of packet latencies. The reason is that the user satisfaction cannot be increased anymore by purely increasing the maximum throughput; the respective values are already beyond the demands of common users. Consequently, the reduction of latency is more rewarding at the present stage.

1.1 Motivation

Latency measurements have been straight forward in the old days; a connection between communication partners (hosts) was often composed of symmetric links with deterministic latencies. In such a scenario it is sufficient to send one packet from one host to the other (request) and wait for the respective reply. The time interval between departure of the request and arrival of the reply corresponds to twice the one-way delay. This procedure is commonly known as *ping* (named after the respective software tool).
Nowadays, communication networks are neither symmetric nor deterministic. They are composed of several links which may be wired or wireless, including intelligent network components which **react** on the traffic they have to deal with. Those are governed by complex link control algorithms, which schedule packet transmission times and distribute the available link capacity among several users. Consequently, a simple *ping* is by far not enough to determine the latency of this connection.
Advanced concepts have to be applied in this case, reaching from synchronization of measurement nodes to stochastic analyses of the measurement results. Generally speaking, latency measurement methodologies have to keep up with the fast evolution of networks, in order to deliver reliable, precise and reproducible results.

1.2 Outlook and Contributions

This thesis is focused on delay measurements in mobile cellular networks. Known methodologies are verified for their applicability to these networks; whereas open issues and common pitfalls are identified. A comprehensive latency measurement methodology was designed and, further, used to carry out measurements for selected mobile cellular networks. The achieved insights are presented in the following structure:

- The first part of Chapter 2, "*Delay in Communications*", provides a general overview on packet delay in communication networks. Basic definitions are provided, whereas supporting material can be found in Appendix A, "*Protocol Descriptions*". In the second part of the chapter, the central definition of delay is given. It relies on the definitions developed in [1] [2] [3] [4]. The last part of the chapter presents two modeling approaches for delay, (i) the queueing theoretic approach (i.e., for non-reactive networks) and (ii) a novel approach taking reactivity of the network into account. This second approach has been presented in [4] [5].

- Chapter 3, "*Latency Measurements*", explains further issues related to delay measurements. In Section 3.1 several approaches for timekeeping are outlined. How to achieve accurate clock synchronization via Global Positioning System (GPS) is described in detail in Appendix C, "*Timekeeping on Desktop PCs*", which is based on [6]. Section 3.2 highlights secondary issues regarding (i) active and passive measurement strategies, (ii) legal considerations and (iii) hardware requirements. Section 3.3 presents a generic approach for the delay measurement design. It consists of a checklist for the successful accomplishment of the measurement procedure; the respective content is bases on [4] [5]. Section 3.4 provides details about the measurement setup used to obtain the delay measurements presented in the following chapters. It is based on [1] [2] [3] [5].

- Chapter 4, "*Benchmarking Wireless Networks*", presents measurement results from several wireless networks. Section 4.1 shows how the injected traffic patterns influence delays in reactive networks; with supporting material in Appendix D, "*Benchmarking: Field Trial*", which includes results obtained from several cellular networks in Vienna. This content has been published in [7]. Section 4.2 dissects the delay introduced by High Speed Packet Access (HSPA) networks into contributions of single network components; this material has been published in [2].

- Chapter 5, "*Traffic Models*", outlines traffic models for the synthetic generation of network traffic. Thereby, Section 5.1 presents a generic approach for traffic modeling based on Transformed Auto-Regressive Moving-Average (TARMA) models, which is based on [8]. Supporting material is provided in Appendix E, "*Derivations of TARMA Expressions*", with the derivation of analytic expressions; in Appendix F, "*ARMA Models for LRD Series*", with a novel fitting approach for Auto-Regressive Moving-Average (ARMA) parameters; and in Appendix G, "*TARMA Application Models*", with parameters of TARMA models for specific applications. Further, traffic traces of online games have been captured and parametrized in this context, which have been noted by the 3rd Generation Partnership Project (3GPP) consortium and recorded in TR 36.822 [9]. Section 5.2 shows how Machine-to-Machine Communication (M2M) traffic can be modeled. The respective content is based on [10]. Section 5.3 explains how cellular background traffic can be modeled realistically; it is based on [11].

- Finally, Chapter 6, "*Conclusion*", summarizes and concludes this thesis.

1.3 Related Publications

The present work is based on experiences obtained from several projects, namely, the *LOLA*[1] project, as well as, the *DARWIN++*, *DARWIN 3* and *DARWIN 4* projects[2]. The material contained in this thesis has partly been published in the respective reports, which can be found on the project websites.
The following publications contain material which originated from these projects; hence, they partly anticipate presented results.

[1] M. Laner, P. Svoboda, and M. Rupp, "Dissecting 3G Uplink Delay by Measuring in an Operational HSPA Network," in *PAM'11, Atlanta, Georgia*, 2011

[2] M. Laner, P. Svoboda, and M. Rupp, "Latency Analysis of 3G Network Components," in *EW'12, Poznan, Poland*, 2012

[3] 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 *WiNMee'12, Paderborn, Germany*, 2012

[4] P. Svoboda, M. Laner, J. Fabini, M. Rupp, and F. Ricciato, "Packet Delay Measurements in Reactive IP Networks," *IEEE Instrum. Meas. Mag.*, vol. 15(6), pp. 36–43, 2012

[5] M. Laner, J. Fabini, P. Svoboda, and M. Rupp, "End-to-end Delay in Mobile Networks: Does the Traffic Pattern Matter?," in *ISWCS'13, Ilmenau, Germany*, 2013

[6] M. Laner, S. Caban, P. Svoboda, and M. Rupp, "Time Synchronization Performance of Desktop Computers," in *ISPCS'11, Munich*, 2011

[7] M. Laner, P. Svoboda, and M. Rupp, "A Benchmark Methodology For End-to-End Delay of Reactive Mobile Networks," in *WD'13, Valencia, Spain*, 2013

[8] M. Laner, P. Svoboda, and M. Rupp, "Modeling Randomness in Network Traffic," in *SIGMETRICS'12, London, UK*, 2012

[9] 3GPP, "TR 36.822, LTE RAN enhancements for diverse data applications." [Online]. Available: `http://www.3gpp.org/ftp/Specs/html-info/36822.htm`

[10] M. Laner, P. Svoboda, N. Nikaein, and M. Rupp, "Traffic Models for Machine Type Communications," in *ISWCS'13, Ilmenau, Germany*, 2013

[11] M. Laner, P. Svoboda, S. Schwarz, and M. Rupp, "Users in Cells: a Data Traffic Analysis," in *WCNC'12, Paris, France*, 2012

[12] EU FP 7 LOLA Project, "Work Package 3, Traffic Measurement and Modelling, Deliverables 3.1–3.6," 2013. [Online]. Available: `http://www.ict-lola.eu/deliverables/wp3-traffic-measurement-and-modelling`

[1] `http://www.ict-lola.eu`
[2] `http://www.ftw.at`

Chapter 2

Delay in Communications

Information exchanged between two remote hosts (source and destination) necessarily experiences delay. It is lower bounded by the distance between the nodes divided by the speed of light. The term *delay* denotes the difference in time between two events, namely, (i) the availability of the data for transmission at node A (e.g., the source) and (ii) complete reception of the data at node B (e.g., the destination). In the following chapter, I present the basic concept of delay in packet based communication networks. The focus is thereby on single delay values experienced for specific datagrams between two nodes. This constitutes the basis for extended concepts at the end of this chapter, concerning (i) sequences of links and routers, i.e., network paths and (ii) sequences of datagrams, i.e., data-streams. Throughout the text the terms *latency* and *delay* are used interchangeably.

2.1 Data Communication Basics

In the above definition of latency several aspects remain unclear. First, what is *data* or *information*? Second, what are *source*, *destination* and other *network nodes*? This section explains the main principles of information exchange over a network. It introduces the network elements and protocols and clarifies on the respective interaction; hence, lays the foundation for the definition of delay provided in the next section.

2.1.1 Definitions of Network Elements

Communication networks consists of two types of abstract elements: *nodes* and *links*. A **node** is a concentrated physical device able to manipulate signals which contain information about the data. Examples are: source host, destination host, routers, switches, bridges and repeaters. A **link** is a distributed medium deployed for the exchange of signals between two or more nodes. For example: cables, fibers, air and vacuum. A **route** is a sequence of connected nodes and links of a network which describes the elements visited by a packet (i.e., layer 3 datagram, explained in Section 2.1.2) on its way from the source to the destination. The type of entries is alternating between links and nodes. An example is the route depicted in Figure 2.1, which can be written as: (Source, Link 1, Node A, Link 2, Node B, Link 3, Destination). Within this thesis, I couple the term route to packets and reserve it for the entire sequence of nodes and links between source and destination; hence, a route always starts with the source host and ends with the destination host. This must not be confused with the term **path**, which denotes any connected sequence of nodes and links within a network, starting and ending with nodes (e.g., [13]). A **route section** is a connected subset of a route. It may start and

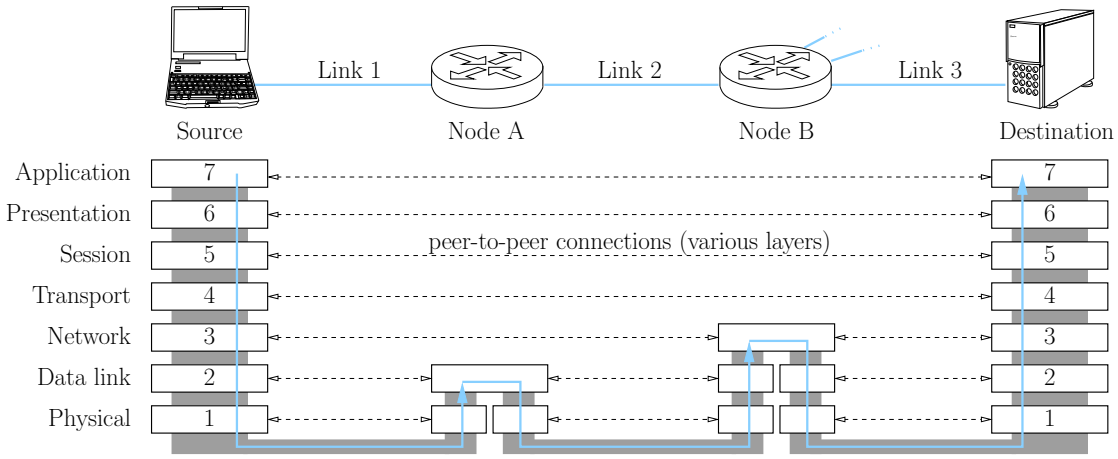


FIGURE 2.1: Interaction between OSI layers and the forwarding of information

end with either links or nodes, however, it never contains both source and destination. An example for a route section from Figure 2.1 could be (Node B, Link 3, Destination). A special type of route section is a **hop**, it consists of one pair of link and node, e.g., (Link 1, Node A) in Figure 2.1.

2.1.2 Protocol Stacks

Packetized networking is based on a combination of multiple tasks, sectioned into **layers**, which rely on each other. Each layer has its well defined functionality, provides services to the upper layer and uses services provided by the lower layer [14]. This is comparable to communication through postal mail. Thereby, the writer of a letter puts it into an envelope and brings it to the postal office, relying on the respective delivery service. The postal office itself uses the service provided by carriers, which physically deliver the letter to the destination office. In terms of computer networking the International Organization for Standardization (ISO) defined a unified framework called Open System Interconnection (OSI) in the late 1970s [15]. The purpose is to provide a common basis (reference model) for standard development and system interconnection.
The OSI reference model defines seven layers with distinct tasks, see Table 2.1. Each of the layers defines end-points which communicate according to the respective rules; as depicted in Figure 2.1. The end-points of the upper four layers are software instances located at source and destination hosts, whereas the end-points of the lower three layers are physical devices and can be located at each node on the communication route. Interfaces between the adjacent layers enable the passing of data down the protocol

	Layer	Functionality	Example
7	Application	User interfaces and support for services	HTTP, FTP
6	Presentation	Syntax and semantics, encryption, compression	ASN.1, XML
5	Session	Dialog control, application synchronization	RTP, TLS/SSL
4	Transport	Service point addressing, connection control	TCP, UDP
3	Networking	Logical addressing (host), routing	IP
2	Data link	Physical addressing, framing, access control	ATM, PPP
1	Physical	Defining bits, physical interface	Ethernet, WLAN

TABLE 2.1: The seven OSI layers defined in [15].

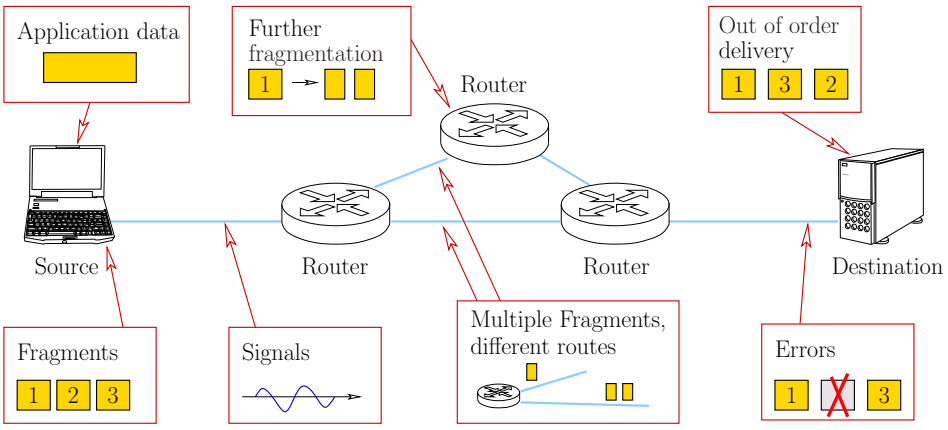


FIGURE 2.2: Several issues which have to be considered when measuring delay.

stack at the source and up at the destination. In parallel also control information can be forwarded. Each layer defines two containers for the data: the **payload** or Service Data Unit (SDU) which is exchanged with the upper layer and the **datagram** or Protocol Data Unit (PDU) for the communication with the lower layer. When the end-point of a specific layer receives payload from the upper layer, this data is encapsulated in one or more datagrams (in order to enable the proper fulfillment of the assigned task) and passed on to the lower layer. Depending on the layer, the encapsulation may comprise: (i) attachment of a header and trailer to the payload, as well as (ii) modification and manipulation of the payload. On the reception of the datagram from the lower layer at the destination end-point, the payload is reconstructed and passed on to the upper layer. Accordingly, the payload (SDU) of the lower layer is equivalent to the datagram (PDU) of the upper layer. This highly modular concept allows for any arbitrary modification of algorithms and procedures inside a single layer. As long as the given functionality is provided to the upper layers, they are not affected by changes in lower layers. For the lowest two layers this concept is of particular importance, since the respective protocols may even change along the communication route between source and destination. Referring for example to Figure 2.1, the physical layer protocol on Link 1 could be that of WLAN, whereas the physical layer protocol on Link 2 is the one defined by Ethernet.

2.2 Definition of Delay

The general mechanisms of networking given above shed light on the conditions under which delay shall be measured and modeled. It remains to find a general and flexible definition of latency. In the following, three questions are carefully examined: (i) What shall be defined as the *data-unit* experiencing delay or, in other words, which protocol layer shall supply this data-unit? (ii) Which *measurement points* or which network elements on a route are adequate for the generation of events? (iii) What should be defined as *event* for the calculation of the delay metric?
These questions are motivated by some challenges encountered when measuring delay; respective examples are highlighted in Figure 2.2: (i) information appears in different formats along the route, e.g., raw data, packets or signals, (ii) packets are possibly delivered erroneously or (iii) information may be fragmented and packets could arrive out of order. A solid definition of delay has to take several of these cases into account.

In this section I present the definition of delay, used for the rest of this thesis. This definition is partly overlapping with definitions known from literature and inherits concepts provided by diverse standardization organizations. Therefore, the first part of this sections reviews existing literature, reveals the sources of borrowed concepts and highlights respective differences.

2.2.1 Related Work on Delay Definitions

Several definitions of delay exist in literature; however, they have a somewhat diverging understanding of *data-units*, *measurement points* and *events*. The reason is that the fields of application are different.
The definition of latency, provided later in this chapter (cf. Section 2.2.4) and used in the rest of this thesis, is a consolidation of various well-known concepts provided by several standardization organizations. Thereby, it is required to give an own definition of delay, in order to have a coherent definition which matches all problem statements treated in the rest of this thesis. An overview of the most popular definitions of delay is given in the following, including a respective comparison to my definition, provided in Section 2.2.4.

2.2.1.1 IETF

The Internet Engineering Task Force (IETF) is a standardization body which develops Internet standards, such as the Internet protocol suite. With the Internet Protocol Performance Metrics (IPPM) framework, the IETF attempts to standardize measurement and performance evaluation procedures; targeting a unified evaluation framework for diverse performance metrics in arbitrary networks. Since the initiation in the late 1990s, several Request For Comments (RFCs) have been released under the IPPM (32 by now), which are subject to continuous updates and extensions[1].
The central document for the IPPM is RFC 2330 [13], which establishes a general framework for performance assessment in the Internet. It introduces a long list of general terminology for: (i) networks, (ii) issues related to timekeeping, (iii) performance metrics and (iv) measurement errors and uncertainties. Also statistical aspects are discussed in the context of sampling, performance evaluation and testing. The document defines the term *metric* as a measurable quantity related to the performance or reliability of a network; the actual definition of metrics is however left for supporting RFCs. Analogously, a notion of packet is defined as *packet of type P*, being a generic qualifier for any sort of Internet Protocol (IP) packet. This naming convention is intended as reminder for the need of the explicit definition of the exact payload structure of the IP packet when used in relation with specific metrics. A concept similar to the measurement points defined in Section 2.2.3 is not specified in RFC 2330. On the contrary, it is explicitly encouraged to perform measurements within a host, cf. [13, p. 18]. Nevertheless, a possible temporal discrepancy between packets appearing (i) inside a host and (ii) at the interface between host and link is mentioned; this is termed *wire time*. An important aspect for delay measurements is the sampling methodology. RFC 2330 suggests Poisson sampling (or geometric sampling for discrete events) for this purpose. The reason is that Poisson samples are unpredictable and resistant against synchronization effects, cf. Section 2.3.1.4.
Based on the described framework several metrics have been defined by the IPPM working group. Two metrics closely related to latency are: (i) the *one-way delay* metric

[1] datatracker.ietf.org/wg/ippm

defined in RFC 2679 [16] and (ii) the *round-trip delay* metric defined in RFC 2681 [17]. They define the delay of packets between source host and destination host (commonly known as One-Way Delay (OWD)) and the delay of packets traveling from the source to the destination and back (commonly known as Round-Trip Time (RTT)), respectively. The notion of packet refers to an IP PDU; the respective structure is left as open parameter to be reported when performing measurements. Fragmented packets require a reconstruction at the destination and trigger an event after successful fulfillment of this task, otherwise they are declared as lost. For duplicated packets the timestamp of the first duplicate is regarded as event for the calculation of latency. Lost packets are declared to cause undefined delay values (informally, infinite). The sending event is triggered when the first bit of the packet crosses the interface between the source host and the first link in the route. Similarly, the receiving event is triggered when the last bit of the packet is available at the interface between last link and destination host. Note, that this definition does not satisfy the additivity property, cf. Definition 2.1, Section 2.2.4. Both documents describe in detail what kind of uncertainties and errors are expected, especially regarding to wire-times and free running clocks. Samples of the metrics are defined according to the Poisson sampling procedure outlined in RFC 2330. A detailed instruction on how to report samples is provided. The IP packet size (length), an important parameter in this context, is for both metrics fixed to one arbitrary number, restricting all packets to the same size.
A metric for *IP delay variation* is introduced in RFC 3393 [18]. This metric is similar to the term *jitter*, and denotes the difference in OWD of two specified packets. The definition of OWD is thereby analogous to RFC 2679. The motivation for the introduction of this metric is the corresponding sensitivity of real-time applications. Further, delay variation is more robust with respect to timing offsets than ordinary OWD. In contrast to OWD and RTT as defined in the RFCs mentioned above, RFC 3393 introduces two concepts which are further incorporated into subsequent RFCs: (i) measurement points are introduced as spatial coordinates of an event and (ii) the size of the corresponding packet must be reported for each delay value, although the packet streams for sampling are recommended to consist of equally sized packets.
Another delay metric for IPPM is introduced in RFC 3432 [19], namely, *one-way delay for periodic streams*. This document provides extensions to the three metrics introduced above, which only define Poisson streams for the respective sampling. Although periodic streams suffer from diverse shortcomings (cf. RFC 2330), they are introduced because: (i) streaming and multimedia applications exhibit periodic packet streams and (ii) periodic sampling strongly simplifies frequency domain analysis. The metric is based on the definition of OWD provided in RFC 2679 and incorporates the concepts of packet sizes and measurements points similar to RFC 3393. Additionally, multiple protocol layers (layer 2–4) are considered for evaluation. They may yield ancillary information valuable for the determination of the experienced Quality of Service (QoS), which is especially important for streaming applications.
Some shortcoming of the present definition of delay are listed in the following. They are considered in a revised definition which I propose in Section 2.2.4.

- The mentioned RFCs only consider end-to-end delay.
- According to the delay definitions in IPPM the additivity property is not satisfied, cf. Definition 2.1. This prevents the generalization of the definitions of latency to route sections.
- Measurements points (cf. Section 2.2.3) are not explicitly defined in the IPPM framework, instead the notion of wire-time is used.

- The mentioned RFCs define IP PDUs to experience latency, in this work the IP SDU is considered as data-unit of interest.
- An explicit handling of multiple timestamps per data-unit and measurement point is not considered by IPPM.
- Lost packets cause undefined delay values, according to the mentioned RFCs; whereas they do not cause delay values according to the definition in Section 2.2.4.

2.2.1.2 ITU

The International Telecommunication Union (ITU) is an agency of the United Nations with focus on telecommunications. It consists of three divisions, whereas ITU-T is the standardization division. ITU-T released Recommendation I.380 [20] in 1999, entitled *IP packet transfer and availability performance parameters*. This document defines several performance metrics, including delay and delay variation.
In contrast to the IPPM this document concentrates on the pure definition of *performance parameters* (equivalent to the term metric within this text). It is less exhaustive regarding related aspects such as timekeeping, sampling strategies and measurement reporting. Similar to the IPPM standards, IP PDUs are defined as data-units of interest. I.380 gives a profound analysis of the concept of *measurement points*, mainly consistent with the definition given in Section 2.2.3, which is inspired by this document. It defines multiple network elements such as *network sections* and *circuit sections* and respective properties such as *measurable*; implying that the element is bounded by two measurement points.
Four different packet transfers reference events are established: (i) IP packet entry event into a host, (ii) IP packet exit event from a host, (iii) IP packet ingress event into a basic section and (iv) IP packet egress event from a basic section. A timestamp is associated to each event, corresponding to the first or last bit of the packet crossing a measurement point for egress and ingress events, respectively. Further, four packet transfer outcomes are defined: (i) successful, (ii) errored, (iii) spurious and (iv) lost. The term *spurious* denotes packets which have never been sent but are received and a *lost* packet corresponds to a pending delivery of an IP packet after a certain maximum transmission time.
The *IP packet transfer delay* conforms to the temporal difference of ingress and egress events at a measurable section with *successful* or *errored* transfer outcome. The fact that erroneous packets can cause valid delay values is thereby in strong contrast to the definitions of latency given in both IPPM and Section 2.2.4. The possibility of multiple events per packet is not considered.
Shortcomings of this ITU recommendation are listed below and considered in the revised definition of delay given in Section 2.2.4.

- According to the delay definitions in I.380 the additivity property is not satisfied, cf. Definition 2.1. Latency values from route sections can only be added if either only ingress or only egress events are considered.
- I.380 defines IP PDUs to experience latency, in this work the IP SDU is considered as data-unit of interest.
- I.380 enables erroneous packets to cause valid delay values.
- An explicit handling of multiple timestamps per data-unit and measurement point is not considered by the ITU recommendation.

2.2.1.3 3GPP

3rd Generation Partnership Project (3GPP) is a consortium of multiple national and international telecommunication organizations, founded in 1998, with the scope of designing a mobile cellular telecommunication system based on the Global System for Mobile Communications (GSM) standard. The scope was further enlarged to the development and maintenance of GSM, Wide-band Code Division Multiple Access (WCDMA) and Long Term Evolution (LTE). In the context of the definition of requirements for LTE, 3GPP released the document TS 25.913 [21], entitled *Requirements for Evolved UTRA and Evolved UTRAN*, which includes two definitions of latency. The purpose of those definitions is to give essential requirements for a communication technology. This stays in contrast to all other mentioned definitions, which are intended as quality metric. The 3GPP definition is much more specialized. Nevertheless, a review of the definition is given, since it is tailored to mobile networks, which are a central part of the present thesis.
The named document defines *control-plane* latency and *user-plane* latency. The first term refers to the experienced latency between the attempt to transmit data, given the user equipment is idle, and the actual transmission of the data. This value is mainly dominated by the access procedure and the radio connection setup. The mismatch between this definition and the definitions given in the previous sections indicates the broad difference between the targeted networks: Wireless networks inherently handle data as a stream of packets rather than single packets. Control-plane latency is however not within the focus of this thesis.
User-plane latency, on the other hand, is a metric comparable to the other definitions given within this section. It is defined as the transmission time of an IP packet without payload from the mobile terminal to the edge node between mobile access network and mobile core network or vice-versa. Thereby, the explicit payload size of zero is advantageous for the purpose of a requirement. The delay experienced by big packets is influenced by the maximum throughput which, in turn, is influenced by various radio conditions; hence, disadvantageous for a clarity of the requirement. The definition of the measurement points is defined within network nodes, similar to RFC 2330. The explicit circumstances of the measurement of delay (e.g., timekeeping, traffic pattern) are, analogously to I.380, not specified.
Shortcomings of these definitions are listed below and considered in the revised definition of delay given in Section 2.2.4.

- Delay is restricted to one route section (3GPP), i.e., between modem and mobile core network.
- The measurement points are within network nodes.
- 3GPP defines IP PDUs without payload to experience latency, in this work the IP SDU is considered as data-unit of interest.
- Using IP packets of length zero eliminates specifications for the handling of fragments and multiple timestamps per data-unit.
- A reliable connection is provided on the route section [22]; thus, no specifications regarding erroneous delivery, duplicate packets or spurious packet have to be provided.

2.2.2 Definition of Data-Unit

Since there are several definitions of delay provided in literature, which capture different aspects and are not always consistent (cf. Section 2.2.1), I provide a definition of delay tailored to this thesis in the following sections.
Latency strongly depends on the "piece of information" defined to experience latency, which I denote *data-unit*. According to the OSI model described above, the datagrams of each layer qualify for this purpose. On the one hand, it is advantageous to deploy higher layer datagrams, in order to guarantee their integrity over long communication routes (compared to specific links for lower layers). On the other hand, lower layers are favorable since the data which can be transmitted is general (instead of specific application data on higher layers). The respective definitions in the related literature diverge, cf. Section 2.2.1.
The best trade-off is the network layer (layer 3). More specifically, **I define the layer 3 payload (SDU) as data-unit experiencing delay**; however, the respective datagrams (PDU, payload and header, possibly segmented) are used for identification. The corresponding range of validity is the entire communication link, from source to destination. Furthermore, layer 3 payload is transparent for any kind of application data. Throughout this work only one type of layer 3 protocol is considered, namely, Internet Protocol Version 4 (IPv4). As layer 4 protocol, supplying the layer 3 payload, User Datagram Protocol (UDP) was deployed in most of the cases, the presented results however are independent from the specific layer 4 protocol. Basic introductions to both protocols are given in Appendix A.
Note, that any large IP datagram may be split into multiple IP fragments (segments). This is the case when the payload exceeds the Maximum Transmission Unit (MTU) of a specific link on the communication route. Fragments are allowed to travel on different routes from the source to the destination, yielding the notion of a data-unit on intermediate links rather fuzzy. Effectively, a data-unit can only be observed on a link when all related IP fragments where transmitted over the same. Further, the calculation of latency requires one event per link (or per node). The appearance of multiple fragments on a specific link belonging to the same data-unit triggers multiple events instead of only one. The handling of such cases is essential for the definition of delay and discussed in Section 2.2.4. Despite the mentioned ambiguities, the deployment of the IP payload as data-unit is advantageous compared to IP fragments. The reasons are:

- The maximum IP payload size is independent of the MTUs of individual links. With a value of 64 kB it is further an order of magnitude bigger than typical values for MTU (e.g., 1500 B).
- IP fragments could be further fragmented at each layer 3 node throughout the entire route. If so, the delay of single fragments cannot be determined uniquely. The delay of the entire payload, on the other hand, can be determined after successful reconstruction.

2.2.3 Definition of Measurement Points

For the assessment of delay it is essential for the data-unit to trigger two timestamped events. In order to trigger an event, the data-unit must pass a defined physical location. In other words, the data-unit must be observed at either a link or inside a node along its route. Both kind of locations are investigated in the following and are found to yield

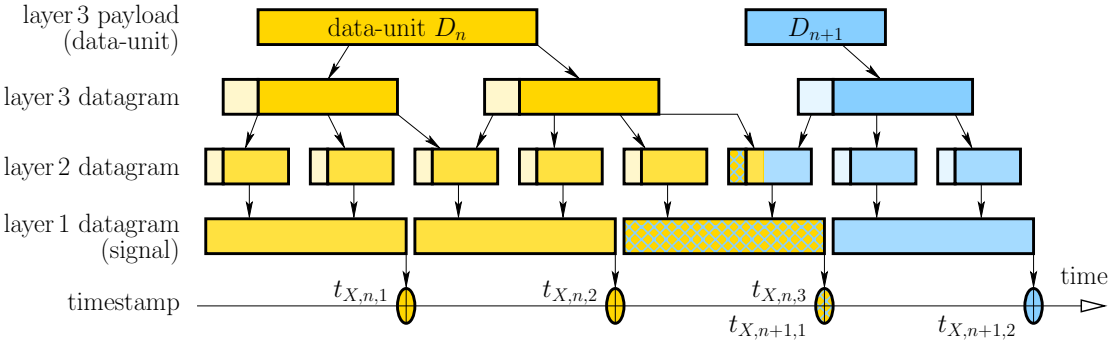


FIGURE 2.3: Assignment of timestamps from interface X to two data-units. Timestamps labeled $t_{X,n,i}$ are assigned to data-unit D_n, those labeled $t_{X,n+1,j}$ to D_{n+1}. Note, that this is not a one-to-one mapping.

some temporal uncertainties of the resulting events. The respective definitions in the related literature diverge, cf. Section 2.2.1.
Network nodes as location for the measurement points have the advantage that the data-units are available in explicit form. The received signals are translated into layer 2 datagrams and, possibly, even layer 3 datagrams are reassembled (i.e., in the case of routers, source host and destination host). The respective accessibility for timestamping purposes, however, is usually limited. The data-unit only appears in memory benches of the network node, which would require dedicated software to perform the timestamping task, i.e., to trigger an event. Thereby two problems arise:

- Any modification of the software of network nodes is critical and may affect the proper functionality of the node. Further, the measurement task could delay the core business of the node, namely, the forwarding of data.
- The hardware architecture of network nodes does usually not support accurate timestamping.

For those reasons we refrain from defining measurement points at the inside of network nodes.
By defining a measurement point on a link, the distributed nature of links has to be taken into account. The exact time an event is triggered depends on the physical location of the measurement point on the link, which is due to the propagation time of signals. Hence, the exact location of the measurement point has to be specified, in order to mitigate any ambiguities. The interfaces between links and nodes are adequate for this purpose and constitute a simple and general solution to this issue. Further, interfaces are easily accessible without requiring any modification to existing hardware.
The data-unit appears on the link in form of signals. The moment of the beginning and ending of a signal can be determined rather precise by the deployment of dedicated hardware. This action is called timestamping (or measuring in the proper sense) and yields a respective timestamp value $t_{X,n,i}$, where X denotes the interface, n the index of the data-unit and i the index of the timestamp. Detailed descriptions of the corresponding measurement setup for several interfaces are provided in Section 3.4. A number of I timestamps may be recorded which can be ascribed to the same data-unit D_n on the same interface X. Further, single signals can carry information on multiple data-units D_n, D_{n+1}, etc. (e.g., due to channel coding) and, consequently, cause multiple identical timestamps assigned to different units $t_{X,n,i} = t_{X,n+1,j} = \cdots$. Single events are therefore blurred over time; there is no one-to-one mapping from a timestamp to an event. An example is given in Figure 2.3, where two data-units generate four timestamps on interface

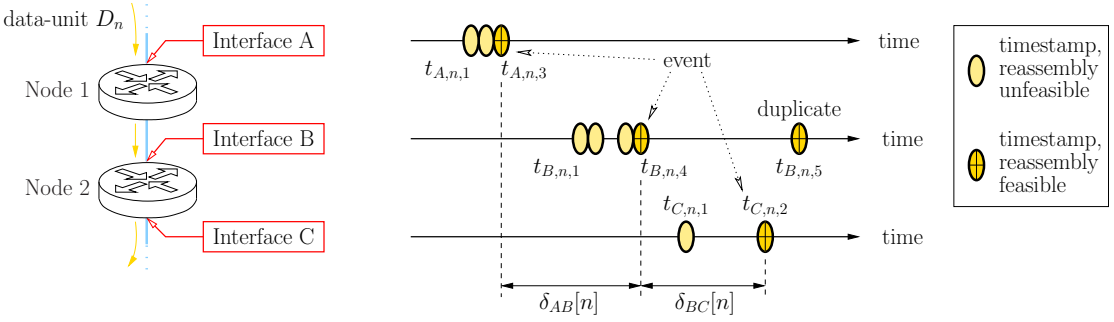


FIGURE 2.4: Calculation of delay over multiple hops. An event is defined as first timestamp from an interface after which correct reassembly of the data-unit is feasible.

X. Thereby three timestamps are assigned to the first data-unit and two timestamps to the second.
As the data-unit crosses an interface, it is not available in its explicit form. This is however required for an assignment of timestamps to data-units. Therefore the data-units have to be reconstructed from recorded signals (decoding, parsing, reassembly). Timestamps are only assigned to a data-unit if the entire data-unit can be reconstructed from the corresponding signals without error, otherwise they are discarded and the packet is declared as lost.

2.2.4 Definition of the Delay Metric[2]

Having defined data-units, measurement points and the timestamping procedure, we can proceed to the definition of *events* and, consequently, the *delay metric*. As central requirement for the delay metric I impose the additivity property, which enables the coherent treatment of latency values over different segments of a route (route sections). I define it as follows:

Definition 2.1 (*Additivity Property*)

Assume two nodes A and C are only connected via the node B. The sum of the delays on successive route sections (A to B) and (B to C) must be equal to the delay on the combined route section (A to C) for each data unit D_n:

$$\delta_{AC}[n] \stackrel{!}{=} \delta_{AB}[n] + \delta_{BC}[n].$$

This requirement seems trivial; however, the most prominent definitions of delay do not comply with it. For example, assume the following definition: Delay is the time between the first bit of the data-unit passing interface X and the last bit of the data-unit passing interface Y. According to this definition, the additivity property in Definition 2.1 is violated, because the sum of the two delay values entails an overlapping interval (i.e., the time between first bit and last bit). This example can be deduced from Figure 2.4, where the first delay values would be the difference between $t_{A,n,1}$ and $t_{B,n,5}$, the second delay value would be calculated from $t_{B,n,1}$ and $t_{C,n,2}$; hence, the interval between $t_{C,n,1}$ and $t_{B,n,5}$ would be counted twice, resulting in a violation of Definition 2.1. This issue can be solved by allowing only one timestamp per interface to be defined as *event*, which is independent of the targeted delay value.

[2]The term metric, often used in this context and defined in [13], has to be understood as synonym for *measurand*. It is not equivalent to a *metric* in the mathematical sense.

Obvious complying definitions would be the first timestamp $t_{X,n,1}$ associated to data-unit D_n or the last timestamp $t_{X,n,I}$, where I denotes the total number of timestamps associated to D_n on interface X. One could even think of constructing artificial timestamps to define an event; for example the mean or median of the I timestamps $t_{X,n,i}$; such constructions are however beyond the scope of this work. The disadvantage of the first timestamp is that it does not consider the serialization time of the packet (i.e., transmission delay, cf. Section 2.3.1.1); hence, the delay would be independent of the packet size and packet retransmissions. The disadvantage of the last packet is that duplicated packets would artificially increase the delay, without contributing any information; in the example exhibited in Figure 2.4 one would even obtain a negative delay value for $\delta_{BC}[n]$.
In order to alleviate the mentioned issues, I define an event as the timestamp $t_{X,n,\iota}$, where ι denotes the index of the first timestamp after which the data-unit can be reconstructed correctly. Consequently, the delay metric can be defined as:

Definition 2.2 (*Delay Metric*)

Assume the data-unit D_n to pass two interfaces X and Y in that order, causing i=1, ..., I timestamps $t_{X,n,i}$ recorded at X and $j = 1, \ldots, J$ timestamps $t_{Y,n,j}$ at Y. Then the delay $\delta_{XY}[n]$ of D_n between interfaces X and Y is defined as

$$\delta_{XY}[n] = t_{Y,n,\iota} - t_{X,n,\upsilon},$$

where ι and υ denote the first indexes after which correct reconstruction of D_n is feasible at the corresponding interface, respectively.

Thereby, the term *correct* implies that the data-unit, equivalent to the IP payload, can be reassembled. If not stated otherwise, this payload corresponds to a UDP datagram, which is tested for correctness deploying the respective checksum; see Appendix A for details. In order to avoid negative delay values, interface X is further required to be closer to the source than interface Y; hence, the data-unit is guaranteed to pass interface X first. In the following the advantages of the presented definition of delay are summarized:

- Delay values on consecutive route sections satisfy the additivity property, as outlined in Definition 2.1.
- The calculated delay value $\delta_{XY}[n]$ is positive, provided D_n passes X before Y.
- The latency includes the wire-time, i.e., components which depend on the maximum possible throughput.
- Duplicates of packets at arbitrary protocol layers are not considered for the calculation of delay.

From the definition of latency in Definition 2.2, two popular special cases can be deduced: (i) OWD and (ii) RTT. Both terms appear often in literature, as outline in Section 2.2.1, the exact specification however varies slightly for different standardization bodies. OWD is usually defined as the delay experienced by D_n traveling from source to destination. This concept is easily adapted to the above definition by defining interface X as the interface between the source host and the first link and, analogously, interface Y as the interface between the last link and the destination of D_n. Some definitions of OWD additionally include the time from the first bit of D_n available for transmission and the last bit of D_n being transmitted at the source. In the present definition, this interval is

	this work	IETF	ITU	3GPP
end-to-end delay (OWD)	✓	✓	✓	
route section delay	all	none	all	one
measurement points at	links	nodes	links	nodes
multiple timestamps per packet	✓			
payload size	0–64 kB	0–1.5 kB	0–1.5 kB	0 B
erroneous packets	dropped	dropped	valid	valid
fragmented packets	valid	dropped	dropped	dropped
lost packets	dropped	valid	valid	dropped
additivity property	✓			

TABLE 2.2: Comparison of delay metrics.

consciously excluded, in order to omit any bias due to the source host. However, this interface is expected to have a large bandwidth (e.g., USB, PCI), yielding the respective interval negligible.
The RTT is often used as latency metric for its ease of implementation in terms of timekeeping; see Section 3.1 for further details. It is an artificial metric, since it involves two data units, D_n and D_m, the first one being a request, traveling from the client to the server, the second one is the response, transmitted from server to client. According to the definition presented in Definition 2.2, RTT can be defined as $\delta_{XX}[n]=\delta_{XY}[n]+\delta_{YX}[m]$, where D_n and D_m are data-units with the same payload, one transmitted after the reception of the other. Again, X and Y are the interfaces between client and first link and server and last link, respectively. However, this definition does not include the server processing time; thus, respective values are biased compared to the conventional definition.
Data-units D_n may trigger no event at an interface X under investigation. Possible reasons are: (i) the data-unit is corrupted, (ii) protocol headers of lower layers are erroneous, (iii) the data-unit or fragments of it traverse the network on a different route or (iv) the events are triggered before or after the observation (measurement) period. For practical reasons a maximum delay $\Delta_{\max}$ is introduced. If D_n does not trigger an event within $\Delta_{\max}$ after its departure from the source host, the data-unit is declared as lost at interface X. If not stated otherwise, this value is assumed to $\Delta_{\max}=255\,\mathrm{s}$, corresponding to the maximum possible time-to-live supported by the IP header; see Appendix A for details. Data-units D_n which trigger no event are explicitly declared as *lost*; hence, a corresponding delay value does not exist and the index n is assigned to the next data-unit. I refrain from assigning an artificial delay value to such data-units (e.g., infinity [16]) since particular statistics could be dominated by those values (e.g., the mean delay is infinity if one delay value from the ensemble is infinity). Instead, a maximum loss ratio $L_{\max}$ is specified for latency measurements, in order to keep the respective impact negligible. Within this work I assume a maximum data-unit loss ratio of $L_{\max}=5\%$.
Finally, a comparison between the all presented definitions of delay is given in Table 2.2.

2.3 Models for Delay

Before conducting latency measurements it is important to presume a tentative model for latency. According to this model the aim of the measurement can be defined; for

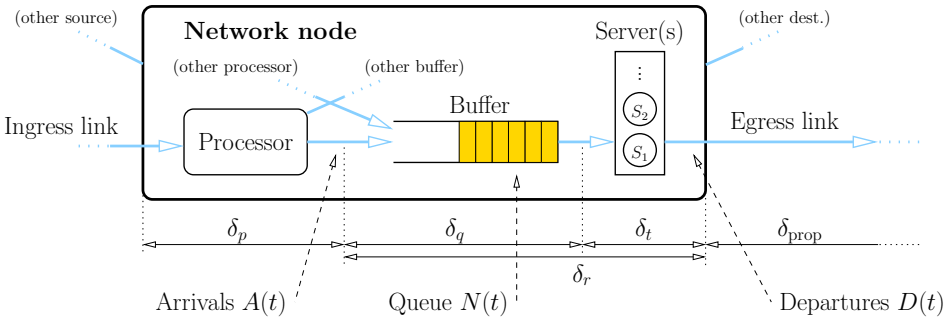


FIGURE 2.5: A network node from the perspective of queueing theory with the four classical delay components.

example, (i) the accurate estimation of parameters of the model, (ii) the validation of hypotheses regarding the model or (iii) the disproof of the model. In light of the desired outcome, the measurement procedure has to be carefully designed. An example is given in Section 2.3.2, where measurements are performed in a cellular network in order to prove its *reactiveness*, based on the conformance with the respective model.
In the following two models for the latency behavior in networks are provided:

- The standard approach based on **queueing theory**.
- A novel model based on the assumption that the network is **reactive**; i.e., it adjusts itself to the input traffic steams of individual users.

2.3.1 Queueing Networks

The classical description of data networks is summarized in the framework of *Queueing theory*. It was initiated by the research of A. K. Erlang, who designed models to describe telephony systems at the beginning of the 20*th* century. At that time the queueing of customers for available communication lines was analyzed; in the present context the customers are substituted with packets, which are buffered within network nodes until they are forwarded. A comprehensive treatment of queueing theory is given in [23] [24]. The detailed interpretation of a network node as queueing system is shown in Figure 2.5 (this could for example be a router). Packets arriving at the ingress interface are decoded by the **processor**, in order to determine the respective destination (egress interface); this element is however not part of a classical queueing system. Afterwards, they are assigned to a **buffer**, where a queue of packets is formed. This is the first central element of the queueing system. The second central element is the **server**, of which multiple instances may exist. The queue is processed by the servers according to the First In – First Out (FIFO) principle. In the context of data communication, the servers can be interpreted as encoders for the physical layer transmission of packets at the egress link. However, also the link itself could be integrated into the notion of a server.

2.3.1.1 Delay in Queueing Networks

The perception of delay in queueing networks is divided into four additive components per hop [23] [25] [26]. The overall per-hop delay can thus be calculated by

$$\delta_{\text{hop}} = \delta_p + \delta_q + \delta_t + \delta_{\text{prop}}, \tag{2.1}$$

with a respective graphical representation in Figure 2.5. The single contributions are thereby:

- **Processing delay** δ_p: This value is the difference in time between a packet arriving at the ingress interface of a network node and the time it is passed to the queue of an outgoing link. The latency is caused by the processor which has to decode the arriving packet and determine the respective link on which it shall be forwarded. It is the counterpart to the server at the egress link; however, without an assigned queue. The respective value is mostly assumed constant, since similar operations have to be performed for any kind of packet.

- **Queueing delay** δ_q is the amount of time a packet is cached (*waiting*) in the buffer before it is treated by the server. In other words it is the time required by the server to process all other packets with higher priority. Consequently, this value is strongly dependent on the cross-traffic, i.e., the overall load of the network component. Accordingly, it is modeled as random variable. In terms of queueing theory, queueing delay is also called **waiting time**.

- **Transmission delay** δ_t corresponds to the time between the beginning of the first signal and the end of the last signal, which are transmitted at the egress interface. This value is caused by the serialization and encoding of the data and is strongly dependent on the overall capacity $C_{\max}$ of the subsequent link. It is equivalent to the length of the time slot assigned to the packet for transmission. Ideally, this value is determined by the linear relation

$$\delta_t[n] = \frac{\pi[n]}{C_{\max}}, \tag{2.2}$$

  where n denotes the packet count and $\pi[n]$ the packet size. In terms of queueing, theory transmission delay is also called **service time**.

- **Propagation delay** δ_{prop}: This latency contribution is defined as the time between the transmission of a signal at the egress interface of the current network node and the respective reception at the ingress interface of the next node on the route. It is caused by the physical distance between network nodes and the maximum speed signals are traveling on the respective link. This may constitute a substantial contribution to the overall latency, especially when considering satellite or transcontinental communication. Due to the mainly constant length of any link between two fixed nodes, this value is usually regarded constant.

Summarizing, the per-hop latency in classical queueing networks consists of: (i) a sum of constant values (processing and propagation delay), (ii) a random value (queueing delay) and (iii) a factor depending linearly on the packet size (transmission delay). Throughout a route, the end-to-end delay (or OWD) is the sum of the contributions of all single hops. This value can potentially be analyzed by the knowledge of the few parameters mentioned above. The most critical contribution is thereby the queueing delay, which has to be handled as random variable. Several results have been established for the latter, which are presented in the rest of this section.

2.3.1.2 Stochastic Processes and Renewal Theory

The customer (packet) arrival process at the queueing system plays a fundamental role for the system performance. Therefore a coarse review of common arrival processes is provided; for further reading refer to [24] [27] [28] [29], in the context of Internet traffic modeling, [30] gives an overview of deployable processes.
The most basic process to be considered in this context is the **binomial process**. Assume discrete time instances t_i and independent Bernoulli trials (coin tosses, possibly

unfair with probability p) performed at each time instant. Arrivals are constituted by positive outcomes of the trial. This simple process has the following properties: (i) independent increments, i.e., the number of arrivals occurring within disjoint intervals is independent, (ii) stationary increments, (iii) geometric distributed inter-arrival times between events and (iv) binomial distribution, i.e., the number of events in disjoint intervals with I time slots [3] is binomial distributed with parameters $\mathcal{B}(I, p)$.
The transition from discrete time to continuous time yields a **Poisson process** from a binomial process. Thereby, the time slots $\epsilon{=}t_{i+1}{-}t_i$ become infinitesimally small, the number of coin tosses (population) grows to infinity $I{\to}\infty$ and the respective probability of a positive outcome goes to zero $p_i{\to}0$ such that $\lim_{I\to\infty} Ip_i{=}\lambda$. The arrival stream of positive outcomes is called a Poisson process with rate λ (cf. [24, pp. 244ff.]). The properties of the process are (note the analogy to the binomial process):

- Independent increments, i.e., the number of arrivals occurring within disjoint periods is independent.
- Stationary increments, i.e., the distribution of the number of arrivals within a period does only depend on the length of the period, but not on the respective absolute time.
- Exponential distributed inter-arrival times and single arrivals, i.e., the Inter Packet-Arrival Time (IAT) $\tau[n]$ between the $(n{-}1)$th and the nth customer (packet) is exponential distributed and arrivals are always of single customers. Further, all IATs are mutually independent.
- Poisson distribution, i.e., consider a period of fixed duration Δt, the number of arrivals within this period is Poisson distributed with parameter $\lambda\,\Delta t$.

Further attractive features of the Poisson process are listed below (cf. [27, pp. 28–29]). Consider a fixed point in time t and a Poisson process with rate λ. The *age* of the last arrival at t (time between last arrival and t) as well as its residual (time between t and next arrival) are independent exponentially distributed random variables with rate λ. Another feature concerns the superposition and splitting of Poisson processes, namely, the superposition of Poisson processes is a Poisson process with a rate equal to the sum of both rates. If a Poisson process is split into two processes (arrival events are randomly assigned to one of the two resulting processes with probability p), then the outcome are two Poisson processes with rate λp and $\lambda(1{-}p)$. Finally, the *Palm–Khintchine theorem* (cf. [24, p. 250]) states that the superposition of a large number of non-Poisson processes with vanishing rate is a Poisson process. In Section 2.3.1.4 and Section 2.3.1.5 it is shown that Poisson arrival processes yield strong results in queueing theory which are largely analytical tractable.
Renewal processes are processes with independent identically distributed (i.i.d.) IATs $\tau[n]$; which is a generalization of the Poisson process (cf. [27, pp. 35ff.]). This is the typical process encountered for the lifetime of some component, which needs constant replacement (i.e., renewal). It can be shown that an arrival rate λ exists in the limit as $t{\to}\infty$. The respective increments (number of arrivals within a period) are neither stationary nor independent in general. However, if a renewal process is constructed such that the first IAT $\tau[1]$ is distributed as the induced residual life, then stationarity can be achieved. Such a process is termed *equilibrium renewal process*. Thus, the Poisson process is the special case where renewal process and equilibrium renewal process

[3] This variable is *not* related to the number of timestamps I introduced in Section 2.2.4.

coincide. Distributions for age and residual life of a equilibrium renewal process can be derived, yielding to closed form solutions in queueing theory, cf. Section 2.3.1.5.
A **Markovian Arrival Process (MAP)** is a process based on a Markov chain [31] [27, pp. 324ff.] (possibly in continuous time). Certain transitions within the Markov chain are marked, connoting the generation of an arrival on the respective transition. This type of processes is a generalization of binomial and Poisson processes, which constitute MAPs with one state and one marked transition in discrete and continuous time, respectively. Many other types of renewal and non-renewal processes are special cases of MAPs, among others: (i) **Markov Modulated Poisson Process (MMPP)**, i.e., a combination of multiple Poisson processes which are sequentially visited according to a underlying (modulating) Markov chain (cf. [29, pp. 65ff.]), MMPPs are non-renewal. (ii) **Phase-type** processes, i.e., a Markov chain with an absorbing state which triggers an event on absorption (renewal process) [32, pp. 24ff.]. Diverse algorithmic results are available for queueing systems fed by MAPs, see [31].

2.3.1.3 Queueing Theory: Basic Definitions

Several properties are known to influence the behavior of a queueing system and the respective delay. Kendall introduced a notation in 1953 [33] which provides a consistent classification of such systems. It consists of three entities, which have been extended since to the following range of factors[4]:

$$A/S/c/k/N/D.$$

Nevertheless, often only the first three factors are used, as originally proposed. The entities have the following meaning:

- **Arrival process** A: A code describing the arrival process at the queue. Typical values are M for a Poisson process, MAP for a Markovian arrival process, D for a degenerate (constant) arrival process or G for a general process.
- **Service time distribution** S: A code describing the service time (or transmission delay) distribution of the server. The values are chosen from the same set as A.
- **Number of servers** c: The number of servers which process the queue in parallel.
- **Buffer length** k: The buffer length or maximum queue length. When the queue has the respective length, then all packets which additionally arrive are dropped (packet loss). This value can be omitted and is set to $k{=}\infty$ in that case.
- **Population** N: The calling population is the total number of costumers including idle ones. In the context of packet networks this number is infinite, since there is no limit on the number of simultaneously arriving packets. This value can be omitted and is assumed to $N{=}\infty$ in that case.
- **Queue's discipline** D: The priority order in which the packets within the queue are served. Typical values are *First In – First Out (FIFO)*, *Last In – First Out (LIFO)*, *Service In Random Order (SIRO)* and *Priority Service (PNPN)*. This value can be omitted, whereby D=FIFO is assumed.

[4]The naming of these factors may conflict with the notation deployed for the rest of this thesis. Nevertheless, it is used within this paragraph for compliance with literature.

For the further treatment of queues, it is convenient to define a time interval $[0, t]$, for which the packet arrivals at the node are specified as $A(t)$, the departures as $D(t)$ and the number of customers residing in the system as $N(t)$. At the beginning of the observation interval the buffer is empty per definition, $N(0)\doteq 0$; yielding the relations $N(t)=A(t)-D(t)$ and $A(t)\geq D(t)$. Further, the average arrival rate until t is defined as $R_A(t)\doteq\frac{A(t)}{t}$ and the average departure rate until t is defined similar. The total time a nth customer spends in the queue is denoted $\delta_r[n]$, named **response time** or **sojourn time**. It is corresponding to the sum of queueing and transmission delay, $\delta_r[n]=\delta_q[n]+\delta_t[n]$, cf. Figure 2.5.
In statistical terms, the **arrival rate** λ is defined as $\lambda=E\{R_A\}$, where $E\{\cdot\}$ is the expected value and R_A the converging value of $R_A(t)$ as $t\to\infty$. Further, service time δ_t is generally interpreted as random variable with **service rate** $\mu=\frac{c}{E\{\delta_t\}}$. The **utilization**[5] U of the queueing system is the fraction of time the server(s) are busy on average, or equivalently, it equals the expected number of customers in service per server; thus $U \leq 1$. If the utilization reaches one, the queue is said to be *saturated*. In such cases the length of the queue grows to infinity.

2.3.1.4 Palm Calculus

Palm Calculus [28] summarizes a number of results concerning the interdependence of probabilistic laws which are defined from different *points of view*. Consider, for example, a sampling point process (probing process) T_n which is deployed to probe a time-continuous process $X(t)$; The probability that $X(t)$ assumes a certain value $P\{X(t)\}$ over time is clearly not required to equal the respective probability at the sampling instances $P\{X(T_n)\}$; which is named the *Palm probability* $P_0\{X[n]\}$ with respect to the probing point process. A practical example for the application of Palm calculus in the context of delay estimation is given in [34]. Some central concepts of Palm calculus regarding latency are presented below.
In the context of queueing networks two kinds of averages are defined: (i) the average over time and (ii) the average over customers. This concept is of special importance for measurements, since the values of interest are usually averages over time; however, the measurements have to be performed by injecting probes to the system, which are packets (equivalent to customers). An average taken with respect to time is the *average number of customers* in the system until t. It is calculated by $\overline{N}(t)\doteq\frac{1}{t}\int_0^t N(\zeta)d\zeta$. An average taken with respect to the customers is, for example, the *average response time* until t. It is determined by $\overline{\delta_r}(t)\doteq\frac{1}{A(t)}\sum_{n=1}^{A(t)}\delta_r[n]$. The average values defined so far can be shown to converge almost sure as $t\to\infty$ see [24, pp. 283ff.]; yielding (i) $\overline{N}(t)\xrightarrow{a.s.}\overline{N}$ the average number of customers in the system, (ii) $\overline{\delta_r}(t)\xrightarrow{a.s.}\overline{\delta_r}$ the average response time and (iii) $R_A(t)\xrightarrow{a.s.}\lambda$ the average arrival rate. Further, they are consistent with the respective expectation values $E\{\cdot\}$ in most practical relevant cases, cf. [24, p. 288]: (i) $E\{N\}=\overline{N}$ and (ii) $E\{\delta_r\}=\overline{\delta_r}$.
A theorem, relying on the above definitions, is presented in the following. Consider a queueing system (possibly more general than that depicted in Figure 2.5) with an arrival rate which is less than the maximum service rate, such that none of the internal queues build up (no saturation, stable system). Accordingly, the involved random variables are

[5] Note, that the utilization is often referred to as ρ in literature, which is however avoided here, since this character is reserved for correlations.

stationary and the tree relations established above are valid. In such cases an important result was established in [35], known as:

Theorem 2.3 (*Little's law [35]*)

Assume the mean number of users $\overline{N}$ in a stable system, the mean arrival rate λ and the mean response time $\overline{\delta_r}$, then

$$\overline{N} = \lambda\, \overline{\delta_r} \qquad \textit{or, equivalently,} \qquad E\{N\} = \lambda\, E\{\delta_r\}.$$

It relates the response time (latency) experienced by the customer (packet) to the number of customers and the respective arrival rate. Note, that averages over time are related to averages over packets, which is the scope of Palm calculus. Theorem 2.3 is a valuable tool for extracting delay statistics; e.g., it can be used to calculate the mean per-hop delay according to Eq. (2.1). An important application of Little's law on the server as separate system is the calculation of the utilization according to

$$U = \min\left(1, \tfrac{\lambda}{\mu}\right), \tag{2.3}$$

which links the arrival rate λ to the service rate μ.
Another result often encountered in literature is known as Poisson Arrivals See Time Averages (PASTA) property and introduced by Wolff in [36]. It states that system properties that are probed by a Poisson point process exhibit the same average (over time) than values obtained by probing (at the probing instances). Further, it is shown that equidistant sampling is vulnerable to synchronization effects, yielding an respective estimation bias; hence, equidistant sampling shall be avoided. The claim for superiority of Poisson traffic in network performance measurements has been relaxed in supporting work of the paper mentioned above [37] [38] [39] [40] [41] [42]. Therein it has been shown, that appropriately randomizing the sampling instances is satisfactory. The current consensus is that any sort of non-intrusive (light) probing traffic (in terms of sampling instances and respective spacing) is suited for unbiased latency measurements, if properly randomized (i.e., independent of any effect to be measured). This is often referred to as *mixing* property, which is a stronger form of joint ergodicity of the sampled process and the sampling process. It has been summarized by Baccelli et al. [43] to:

Theorem 2.4 (*PASTA revisited (NIMASTA) [43]*)

Assume a continuous stationary random process $X(t)$, a discrete stationary sampling process T_n and a positive function $f(\cdot)$. If $X(t)$ and T_n are ergodic and at least one of them is mixing, then

$$\lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} f(X(T_n)) = E\{f(X(t))\}.$$

Since the expression on the left-hand side is a sample average and that on the right-hand side a time average, Theorem 2.4 is a typical example for Palm calculus.
The theorem implies that the values of $X(t)$ can be measured without affecting $X(t)$ by the measurement procedure. In the typical latency measurement scenario this is not the case. Delay probes (packet injected for measurement) alter the global load of the system which in turn has an impact on the global and on the measured delay; such measurements are denoted **intrusive** [44]. The PASTA property states that Poisson processes convey unbiased sampling results in such cases. The reason is the *lack of*

anticipation assumption [36]; namely, the fact that for each time instant t the arrival instances in any arbitrary interval after t are independent from the actual state of the system and, especially, from any arrivals prior to t. As counter-example consider the regular injection of sampling packets into a network: the contribution of the previous packets to the system load at the arrival instant of the current packet is similar for all packets; hence, the origin of a measurement bias. Notice, however, that the term *unbiased* refers to the global system load (i.e., probing traffic and cross-traffic), which is a common pitfall. The pure presence of probing traffic is itself the cause of a bias, when system parameters shall be assessed with respect to the cross-traffic only. Accordingly, non-intrusive probing traffic (rare injection of small packets) is the method of choice in this case, which in turn narrows the advantage of Poisson traffic over any other probing pattern conforming to Theorem 2.4.

2.3.1.5 Results for the M/G/1 Queue

An typical example in the context of communications is the $M/G/1$ queue, for which it is possible to derive fundamental results on the delay. The $M/G/1$ queue is driven by an arrival process which is Poisson, indicated by the key M. This is often encountered in networks, since accumulated traffic can be modeled by a Poisson process due to the Palm–Khintchine theorem, cf. Section 2.3.1.2. The service time distribution is general (G) and one server is processing the packets, as described by the trailing *1*. A single server is typical in the context of networking, since links mostly require only one server. The full notation would conform to $M/G/1/\infty/\infty$/FIFO. Thus, all arriving customers (packets) are served, the system itself does not create or destroy them, which is corresponding to an infinite length of the buffer. The calling population is infinite and the queue discipline is FIFO.
For this queue it can be shown, see [24, pp. 295ff.], that the expected values of the waiting time can be calculated to

$$E\{\delta_q\} = \frac{\lambda E\{\delta_t^2\}}{2(1-U)}. \tag{2.4}$$

This result yields closed form solutions for many kinds of service time distributions of G; hence facilitates the calculation of the per-hop delay according to Eq. (2.1). Combined with Little's law (cf. Theorem 2.3), Eq. (2.4) leads to an expression for the mean number of customers in the system, referred to as *Pollaczek–Khinchin formula*. Further, the queue length distribution and the queueing delay distribution can be obtained for a variety of distributions of G in closed form [45, pp. 68ff.].
Hence, assuming a pure queueing network with Poisson arrival processes yields analytically tractable results for the statistics of the delay; for generalizations (e.g., multiple servers) see [45]. Difficulties arise for heavy-tailed distributions, which can be avoided by fitting them to short-tailed distributions (e.g., hyper–exponential [46]). A remaining challenge is to infer on parameters of the network and/or the cross-traffic from delay (and loss) values measured a network. A popular example is the estimation of available capacity (bandwidth) throughout a network route [47] [48] [26] [42] [49]. Such issues are referred to as *inverse problems* [50] [45].

2.3.2 Reactive Networks

Access networks such as mobile networks, might handle user data in terms of flows, persistent connections or sessions. Examples are: (i) 2G/3G/4G networks assign so-called bearers to mobile devices which coordinate any kind of communication; (ii) firewalls or traffic shaping mechanisms within an access network record flow statistics per IP

end-point pair. Each such session is linked to a set of internal parameters which potentially affect the experienced end-to-end latency of packets. These parameters may depend on the actual packet, but further, on the whole history of transmitted and/or received packets within the same session. For example: (i) radio resources occupied by a bearer may be gradually released after short idle periods [22, pp. 268ff.], hence, the inter packet-arrival time influences achievable throughput and latency; (ii) a packet towards a client will be blocked by the firewall if no respective request was sent, thus, it is depending on the previous packet in the opposite direction. It is concluded that reactive network components associate state machines to traffic sessions, cf. [22]; the history of a session influences the associated state machine, which in turn may affect the experienced end-to-end latency of future packets.
This is a fundamental difference to queueing networks, which handle packets of the same session as independent entities. For queueing networks four additive delay components are assumed per hop, cf. Eq. (2.1); the processing delay δ_p and the propagation delay δ_{prop} are constant, whereas the waiting time δ_q and the service time δ_t are random processes. None of those components inherits the notion of user sessions or alike. Assume that latency is probed in a slightly loaded queueing network ($U \ll 1$, light cross-traffic and non-intrusive probing traffic), then both the waiting time process (δ_q) experienced by the probes as well as the service time (δ_t) are uncorrelated (except for controllable correlations caused by the packet sizes of the probing stream). Further, for non-intrusive probing traffic only, any correlations within the delay must spring from δ_q, being caused by cross-traffic. This means that the history of the probing pattern has no influence on the overall delay. This assumption needs to be abandoned for reactive networks.
In the following I give a rigorous definition of reactiveness. A novel model for the delay behavior is proposed, being required to describe the delay behavior observed in mobile cellular networks. This model will be the basis for the design of delay measurement procedures presented in later chapters.

2.3.2.1 Definition of Reactiveness

A network is considered as reactive if at least one node on the data-route is reactive. The definition of reactiveness will be given in terms of the latency response on traffic streams. The latency on an arbitrary route section is modeled as a random process denoted $\delta[n]$, where n is the packet index. Instead of further separating the delay into additive components (e.g., four components per hop in queueing networks), it is perceived as a whole; being satisfactory from the perspective of measurements. A further breakdown can be considered for modeling issues (as discussed at the end of this section); it however requires an accurate specification of the probabilistic relations among components (e.g., possible correlations between delays on separate route sections).
The overall latency is influenced by several factors according to

$$\delta[n] = f(\theta, \psi, \phi), \tag{2.5}$$

where $f(\cdot)$ denotes an arbitrary function, and θ, ψ and ϕ denote representatives of the three independent sets of parameters Θ, Ψ and Φ, respectively. Those are explained in Table 2.3 and shortly summarized in the following (see [4] for an extended list).
The set (vector space) Θ comprises all possible parameters related to the probing traffic and the respective pattern (including the reverse link); whereas θ is a respective representative, i.e., a traffic pattern. It constitutes the main difference to the concept of latency established in queueing theory, namely, the set of parameters of the session history on which a network node reacts. Two types of parameters of Θ are of special interest, i.e.,

Θ	probing traffic (session)
	• past and present packet sizes $\pi[n]$ • past and present inter packet-arrival times $\tau[n]$ • data rate R • variations in the instantaneous data rate $r[n]$ • traffic on reverse link • protocol (e.g., TCP, UDP), port
Ψ	short term variability
	• offset between packet arrivals and time slots • fluctuations in network load (queue lengths) due to cross-traffic • retransmissions and hybrid retransmissions • interference and noise on (wireless) links • fast fading radio channel • measurement uncertainty
Φ	constant or slowly changing effects
	• maximum end-to-end throughput $C_{\max}$ • physical location (e.g., end-to-end distance) • global network workload • time of day (diurnal patterns) • service level agreements • modem, communication technology and provider • measurement bias (caused by the gauge)

TABLE 2.3: Parameter sets influencing the delay.

all past and present Packet Sizes (PSs) and Inter Packet-Arrival Times (IATs) of the probing traffic (pattern). The packet size is denoted $\pi[n]$ and the inter packet-arrival time (idle time *before* the packet) $\tau[n]$. Further, all properties of the nth packet of the pattern θ are summarized in the vector $\theta[n]$.
Communication routes are characterized by an individual maximum end-to-end throughput $C_{\max}$ (or bandwidth, capacity), cf. Section 2.3.1.1. Traffic patterns which approach or exceed this value suffer from non-stationary delay and/or enhanced packet loss. This is equivalent to a system utilization $U \approx 1$, cf. Eq. (2.3). Such effects are not subject to the present investigation. Further, single packets of probing traffic shall not affect each other in a queueing theoretic sense. More precisely: the system load imposed by probing traffic and cross-traffic must be light; thus, the utilization must be low, $U \ll 1$. A detailed investigation on this requirement is given in Appendix B. It shows that the involved queues must be empty between any two consecutive probing packets, at least for a short period of time. Accordingly, the probability P_{NE} that the queue is never empty between two probing packets must be close to zero,

$$P_{NE} \leq \epsilon, \tag{2.6}$$

for a small positive constant ϵ, cf. Appendix B. An overall utilization of $U \leq 0.3$ is satisfactory for this purpose. Accordingly, the instantaneous data rate

$$r[n] = \frac{\pi[n]}{\tau[n]} \tag{2.7}$$

of a probing pattern θ has to be bounded. The probing traffic Θ is therefore a priori constrained to patterns θ, fulfilling the requirement

$$\Theta \doteq \{\theta : r[n] \leq C_{\max} U_p \; \forall n \in \mathbb{N}\}, \tag{2.8}$$

namely, the instantaneous data rate is always smaller than the maximum feasible end-to-end throughput times the maximum utilization U_p caused by the probing traffic. A value of $U_p \leq 0.1$ is satisfactory for most practical issues.
The remaining influence of the maximum end-to-end throughput on the experienced latency is due to the the packet transmission time $\delta_{\mathrm{t}}[n]$. According to Eq. (2.2) this influence is commonly assumed as linearly related to the packet size, cf. Section 2.3.1.1. Consequently, if a probing pattern θ conforms to Eq. (2.8) and $U \ll 1$ for all involved queues, a latency response of

$$\delta[n] = \tfrac{\pi[n]}{C_{\max}} + f'(\psi, \phi) \tag{2.9}$$

can be observed over a route section. Thereby, $f'(\cdot)$ denotes an arbitrary nonnegative relation. Note, that $\pi[n]$ is the only representative of θ appearing in Eq. (2.9). In real life scenarios the transmission delay may experience a non-linear influence from the packet size; for example, due to packet segmentation. This could be interpreted as an immediate reaction of the network node on the traffic, however, an influence of the history of the traffic pattern is still not given. Thus, those route sections are **non-reactive** and adhere to the latency response

$$\delta[n] = f''(\pi[n], \psi, \phi), \tag{2.10}$$

where $f''(\cdot)$ is an arbitrary non-negative and possibly non-linear relation.
In the following the focus is on reactions on the history of the traffic pattern (including the present IAT $\tau[n]$, since it bears information about the last packet). Reactions on the present PS, according to Eq. (2.9) and Eq. (2.10), are therefore explicitly excluded from our investigation. A *reactive* route section is thus defined as a route section for which two traffic patterns yield different latency responses, although the present packet sizes are equal.

Definition 2.5 (*Reactiveness*)

Assume Θ adhering to Eq. (2.8) and $U \ll 1$ for all involved queues, such that Eq. (2.6) holds. A network route section is reactive *if*

$$\exists\, \theta_i, \theta_j \in \Theta, n \in \mathbb{N} \Rightarrow$$
$$\pi_i[n] = \pi_j[n] \;\wedge\; P\{\delta[n] | \theta_i, \psi, \phi\} \neq P\{\delta[n] | \theta_j, \psi, \phi\},$$

where $P\{\cdot\}$ denotes the probability, $\cdot|\cdot$ the conditioning operation and ψ and ϕ arbitrary but fixed samples from Ψ and Φ, respectively.

In this case, the history of the packet stream influences the delay figures, although any influence from queueing effects is explicitly suppressed. In order to emphasize the practical relevance of reactiveness in modern networks, respective examples are given in the next sections.
In order to incorporate the notion of reactiveness into existing delay models (such as presented for queueing networks in Section 2.3.1.1), Eq. (2.1) can be slightly modified. This requires to abolish the assumption that the processing delay δ_p is a constant. It should rather be assumed as random process causing the reactiveness discussed in Definition 2.5. Although this solves the ambiguities between the two models, some popular paradigms of

the queueing theoretic approach have to be abandoned (e.g., non-intrusiveness combined with the PASTA property do *not* yield unbiased samples). This is further illustrated in the examples given below.

2.3.2.2 A Basic Synthetic Example

In this section the notion of reactiveness given in Definition 2.5 is illustrated by a basic example. Despite of its simplicity, the example sufficiently illustrates challenges encountered with ordinary random probing in reactive networks. We adhere to Poisson probing patterns only for the respective practical relevance; the experienced ambiguities would also arise with other kinds of randomized probing traffic.
The basic model represents a network reacting on the IATs of packets within data streams. In practice, such behavior is exhibited by persistent connections in last-mile networks, where network resources are released after a certain idle time T_τ, in order to enable faster communication for other users. Note that this type of reactiveness is very limited; only one parameter of θ (i.e., the IAT $\tau[n]$) influences the respective delay. Assume that IATs above a threshold T_τ=0.3 s cause a constant extra delay of 2 ms for the next transmitted packet; IATs below T_τ do not cause this delay. A random delay component caused by ψ and a fixed offset caused by ϕ constitute an independent contribution $f'''(\psi,\phi) \sim \mathcal{N}(9\,\text{ms}, (1\,\text{ms})^2)$, which is assumed Gaussian distributed with positive mean. These assumptions can be condensed in the delay model

$$\delta[n] = f'''(\psi,\phi) + \begin{cases} 2\,\text{ms}, & \text{if } \tau[n] > T_\tau \\ 0, & \text{otherwise.} \end{cases} \tag{2.11}$$

For the basic model the packet size has no influence on the delay. Consequently, from the queueing theoretic perspective, the transmission delay tends to zero $\delta_t[n]\to 0$, or equivalently, the maximum capacity of the system tends to infinity, $C_{\max}\to\infty$, cf. Eq. (2.2); causing the respective utilization to tend to zero $U\to 0$, cf. Section 2.3.1.3.
By probing the delay we deploy various Poisson processes with different mean packet rates $\overline{\lambda}=\frac{1}{\overline{\tau}}$. The IAT of a Poisson process is exponential distributed, with mean $\overline{\tau}$. The exponential distribution is relatively compact in the sense that more than 90% of its probability mass lies within two decades (the fact that the respective support reaches from zero to infinity is thereby slightly misleading). This is illustrated in Figure 2.6 (a), where exponential Cumulative Distribution Functions (CDFs) of IAT are given for five different mean values $\overline{\tau}$ on logarithmic scale. It is problematic to detect reactions of the network on the probing pattern θ if the pattern does not trigger a statistically significant amount of such reactions. This issue is intensified by fluctuations in $f'''(\psi,\phi)$ which further obscure the influence of θ. Consequently, the mean rate $\overline{\lambda}$ of the Poisson process should be chosen a priori such that the searched reactions are excited to a sufficient extend. However, this would require previous knowledge of the reactions of the network (e.g., Eq. (2.11)), which cannot be assumed in practice.
Simulation results obtained for the outlined example are illustrated in Figure 2.6. Therein the IAT distribution of the probing patterns is depicted at the left-hand side, together with the threshold T_τ which triggers the reaction of the network according to Eq. (2.11). The left-most and right-most probing patterns ($\overline{\tau}$=0.03 s and $\overline{\tau}$=30 s) are both mainly concentrated at one side of the threshold; the respective delay figures, see Figure 2.6 (b), do only show one of the cases outlined in Eq. (2.11). The resulting Empirical Cumulative Distribution Functions (ECDFs) are Gaussian, with a mean of 9 ms and 11 ms; consequently, the complete network behavior cannot be inferred. The three intermediate

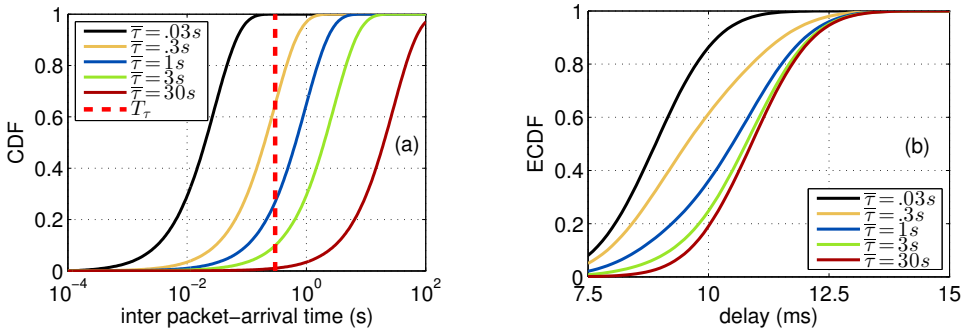


FIGURE 2.6: Network reacting on IATs. (a) exponential IAT distributions of multiple Poisson processes with different rates $\overline{\tau}$. The dashed line indicates the network's threshold T_τ. (b) delay distribution as reaction on different probing patterns, cf. Eq. (2.11).

probing patterns trigger reactions of the network; they excite delay ECDFs being a mixture of the two Gaussian distributions. Thereby it is possible to infer on the full network behavior from the measurements. In other words, Eq. (2.11) can be reconstructed by compiling latency statistics *conditioned* on the IAT. Nevertheless, it is easy to construct examples which are explicitly ill suited for Poisson probing and, accordingly, do not allow for a full reconstruction. For instance, (i) reactions on the mean data rate or (ii) reactions on the fluctuations of the data rate.
The key conclusions drawn from this simple example are:

- In the basic example the maximum throughput is assumed very high, yielding utilizations close to zero; hence, the five probing patterns are non-intrusive. However, all of them cause different delay responses. Consequently, *non-intrusive* probing patterns are not guaranteed to measure meaningful performance metrics. In other words, this popular concept may lead to disastrous misinterpretations if the network is reactive.

- The PASTA property, cf. Theorem 2.4, is satisfied in the basic example, in the sense that time averages equal sample averages. However, the PASTA property is often associated with unbiasedness. In the present example all five latency performance curves shown in Figure 2.6 (b) are different, they are biased by the respective Poisson probing pattern. Thus, in reactive networks the *PASTA property* is not leading to *unbiasedness*.

- The latency in a reactive network is according to Eq. (2.5) depending on parameters of the traffic Θ (including the session history). Any measurement of the delay is conditioned on the respective probing traffic θ_p, yielding $\delta[n]{=}f(\theta_p, \psi, \theta)$. However, one should keep in mind that this may not be the desired outcome and prefer a complete description of the network behavior $\delta[n]{=}f(\theta, \psi, \theta)$ (such as Eq. (2.5)), which is valid for any $\theta{\in}\Theta$.

- Probing patterns can be optimized to trigger specific reactions of the network; however, there is no *perfect* probing pattern stimulating all reactions to a significant extend, such that the network behavior could be reconstructed from a unique measurement. Relying on one single probing pattern entails the risk of undiscovered effects. Therefore, I suggest to deploy multiple probing patterns.

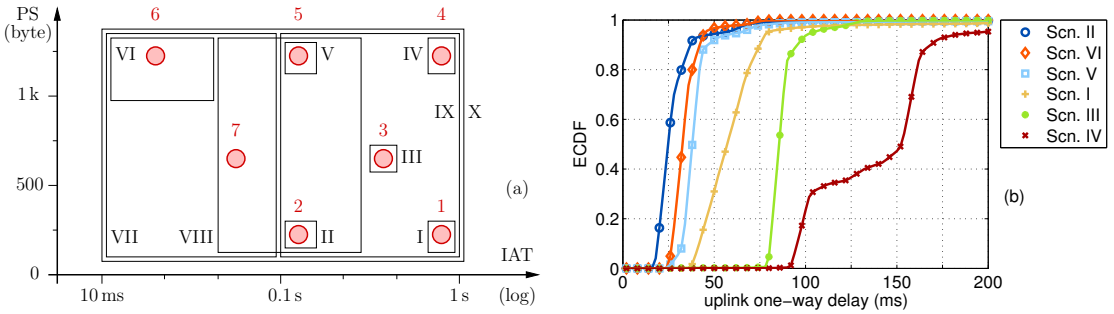


FIGURE 2.7: (a) The measurement scenarios Scenario I–X (black rectangles, roman numerals) are corresponding to PS and IAT distributions of probing patterns. The checkpoints Checkpoint 1–7 (red circles, Arabic numerals) are required to condition the measurement results on a PS-IAT tuple. (b) Measured ECDFs of uplink OWD for an HSPA network, obtained by probing patterns conforming to different scenarios (Scenario I-VI). Huge variations in median latency and jitter are observable.

2.3.2.3 A Real Life Example: HSPA

In order to highlight the practical relevance of reactive networks, I present delay measurement results for High Speed Packet Access (HSPA) in uplink direction. This link type is chosen for illustrative purposes, since the respective OWD exhibits distinguished behavior compared to non-reactive networks. Nevertheless, the discussion could be extended to other types of connections.

Since the scope of this section is to reveal reactiveness in a real operational network, the traffic patterns used for the measurements presented below have been designed in light of Definition 2.5. Accordingly, ten different probing patterns are introduced, denoted Scenario I–X. The respective PSs and IATs are uniform i.i.d. within the limits depicted in Figure 2.7 (a); see [5] for tabulated values. Notice, that several scenarios share regions of the same PS and IAT. Within such overlapping regions I place checkpoints, named Checkpoint 1–7. Latency figures of a specific scenario can be conditioned on a checkpoint; thereby only packets with PS and IAT corresponding to the checkpoint are deployed. By choosing one checkpoint and comparing multiple scenarios at this checkpoint, Definition 2.5 can be validated.

First the latency figures for various scenarios, cf. Figure 2.7 (a), are analyzed on their own. A respective selection is given in Figure 2.7 (b), which shows empirical latency CDFs for Scenario I–VI. Those are non-overlapping scenarios and most compact in the PS-IAT plane. The distances between the single CDFs are enormous, Scenario II yields a median latency of 25 ms, whereas Scenario IV exhibits roughly 150 ms (six times higher). This figure on its own reveals the reactiveness of the network. For example, Scenario I (corresponding to the traffic pattern usually issued by the prominent *ping* program) yields a median latency of more than 50 ms. Changing only the IAT from values around 1 s to 100 ms (corresponding to Scenario II) halves the median latency as well as the latency variation (jitter). This is clearly a reaction of the network on the IAT, since such effects do not occur in pure queueing networks. Conversely, these results give evidence of an influence of the distributions of PS and IAT on the packet latency.

The evaluation method applied above uses compact scenarios in the PS-IAT plane. One could conclude, that deploying a widespread traffic pattern (e.g., Scenario X) and condition the results on certain combinations in the PS-IAT plane (e.g., Checkpoint 1–7) will yield the same latency figures as the above measurements with compact scenarios. This would typically lead to a speedup in the measurement procedure. However, such an assumption would be equivalent to the statement that (i) the resulting latency figures are

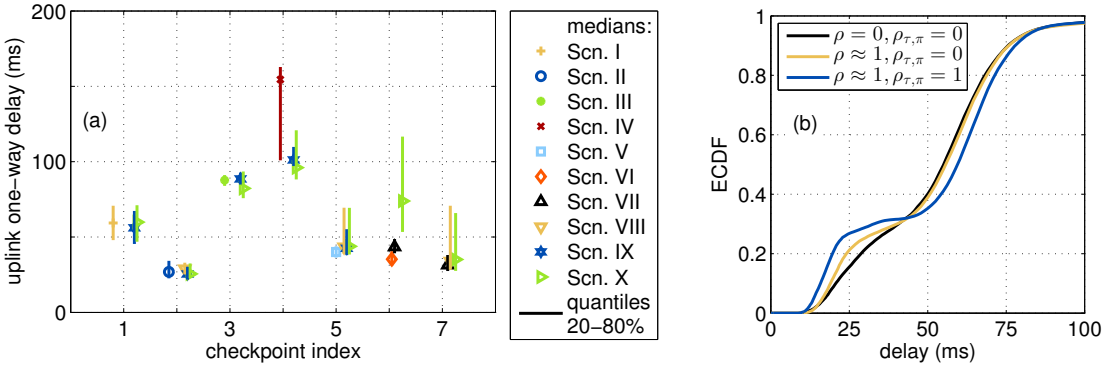


FIGURE 2.8: (a) Comparison of median OWDs for multiple scenarios at different checkpoints. Divergence in the latencies of various scenarios at the one checkpoint indicates the dependence of the latency on the traffic history (observable at Checkpoint 4–7). (b) Influences of autocorrelation and cross-correlation of PS and IAT on the latency ECDF. Distributions of PS and IAT are according to Scenario X in all three cases. Auto-correlation and cross-correlation are gradually increased.

only influenced by the PS and IAT (i.e., $\pi[n]$ and $\tau[n]$) or (ii) the history of the probing pattern (i.e., $\pi[n-m], \tau[n-m], m \in \mathbb{N}$) has no influence on the actual experienced latency $\delta[n]$. Unfortunately, **this is not the case**.

A comparison of latency measurements obtained from various scenarios reveals the real circumstances. For this purpose the seven checkpoints Checkpoint 1–7 have been defined, cf. Figure 2.7 (a). These allow to match the measured latency performance for different scenarios by considering only samples at the checkpoints (corresponding to PS-IAT tuples). This specific selection of samples is equivalent to a conditioning of the latency on PS and IAT. Notice the parallels to Definition 2.5; namely, conditioning on a checkpoint assures that $\pi_i[n]=\pi_j[n]$ and $\tau_i[n]=\tau_j[n]$ for all traffic patterns $\theta_i, \theta_j \in$ Scenario I–X.

The evaluation in Figure 2.8 (a) shows the median delay, as well as the 20th and 80th percentiles, of different scenarios matched at various checkpoints. For certain checkpoints (e.g., Checkpoint 1–3) all scenarios exhibit the same median and spread. On the other hand, there are checkpoints for which the medians are similar but the spread strongly varies (e.g., Checkpoint 5, 7) and, finally, for some checkpoints both the median and the spread are diverging (e.g., Checkpoint 4, 6). The shift of the median latency in the last case amounts to more than 50%. I conclude, that the history of the probing pattern has an influence on the latency, and Definition 2.5 is satisfied; in other words: **this HSPA network is reactive**. This has the inconvenient consequence, that any conditioning should be done not only on the present packet (i.e., $\pi[n]$ and $\tau[n]$), but also on the history (i.e., $\pi[n-m], \tau[n-m], m \in \mathbb{N}$); yielding a high dimensional parameter space.

Another way to confirm the influence of the probing history on the delay is to conduct multiple measurements with the same PS and IAT distributions but different auto-correlation functions. Highly correlated traffic patterns tend to have a history which is close to the actual values of PS and IAT, while this is not the case for uncorrelated patterns. For this purpose I generate the uniform distributed random variables $Z_\pi[n]=Q^{-1}(Y_\pi[n])$ and $Z_\tau[n]=Q^{-1}(Y_\tau[n])$ and map them to Scenario X; where $Y_\pi[n]$ and $Y_\tau[n]$ denote standard normal distributed random processes and $Q^{-1}(\cdot)$ the inverse complementary normal CDF. By introducing an auto-regressive component to the processes, $Y[n]=\varphi \cdot Y[n-1] + X[n]$, it is possible to vary the auto-correlation $\rho[1] \equiv \rho$ from -1 to 1. Further, cross-correlations $\rho_{\pi,\tau}[0] \equiv \rho_{\pi,\tau}$ are introduced between PS and IAT, by

linearly relating the Gaussian processes $X_\pi[n]=\varphi' \cdot X_\tau[n]+X'[n]$. $X[n]$ and $X'[n]$ both denote normal i.i.d random processes. Also $\rho_{\pi,\tau}$ can hence be varied from -1 to 1.
The evaluation of respective measurements is given in Figure 2.8 (b), with PS and IAT distributions according to Scenario X. Three cases are presented: (i) The ordinary Scenario X setup ($\rho=0, \rho_{\pi,\tau}=0$) with fully independent processes. (ii) Strong auto-correlation of both processes ($\rho\approx1, \rho_{\pi,\tau}=0$), achieved by $\varphi=0.99$. (iii) Strong auto-correlation and cross-correlation ($\rho\approx1, \rho_{\pi,\tau}=1$), with $\varphi=0.99, \varphi'=1$. Figure 2.8 (b) reveals noticeable differences between the three latency ECDFs. Note that these ECDFs are averages over the wide area of Scenario X (broad distributions); thus, differences may strongly increase for unlucky choices of more compact distributions of PS and IAT. Therefore, both the auto-correlation and the cross-correlation of the probing pattern influence the outcome of respective latency measurements. Consequently, the history of the traffic pattern is evidenced to influence latency measurements.

Chapter 3

Latency Measurements

In theory the definition of latency, given in the previous chapter, should suffice to perform respective measurements. In real life scenarios, however, delay assessments are fundamentally influenced by non-trivial issues regarding the measurement setup: (i) Delay measurements require time synchronization among measurement probes. (ii) Data-units which experience delay must be injected and captured on the route section under investigation. (iii) The required measurement design must handle influences from a high-dimensional parameter space (cf. Table 2.3), requiring sophisticated pre-experimental planning and a solid measurement design. In this chapter, I give an overview on those challenges.

3.1 Timekeeping

A central aspect of latency measurements is the accurate timestamping of events. This implies that the clocks, performing the timestamping procedure at remote locations, report a common notion of time. A general notion of time (timescale) is defined by ITU-R recommendation TF.460 [51], which defines the Coordinated Universal Time (UTC). This timescale is equivalent to the mean solar time, being a successor to Greenwich Mean Time (GMT). It is based on the frequency of International Atomic Time (TAI), emitted by several time laboratories, and has additional leap seconds introduced at irregular intervals to compensate for the slowing of the rotation of the Earth.
An extensive list of terms related to timekeeping are introduced in RFC 5905 [52], which introduces the Network Time Protocol (NTP) and is one of the central documents handling timekeeping in the Internet. Within this work I adhere to the definitions of RFC 2330 [13], since they are tailored to network measurements and focus on the short-term accuracy. The following terms are thereby defined:

- **Offset** $\Delta T(t_c)$ is the difference between the time reported by a clock $T(t_c)$ and the corresponding UTC time $T_{\mathrm{UTC}}(t_c)$; $\Delta T(t_c) = T(t_c) - T_{\mathrm{UTC}}(t_c)$.
- A clock is **accurate** at a specific moment t_c if its offset is zero, or equivalently, $T(t_c) = T_{\mathrm{UTC}}(t_c)$.
- The **skew** $\Delta f(t_c)$ of a clock at a particular moment is the frequency difference between the clock and UTC. This is equivalent to the first derivative of the offset, $\Delta f(t_c) = \frac{d}{dt_c}\Delta T(t_c)$.
- **Drift** $D(t_c)$ is the variation of the skew. It is the second derivative of the offset, $D(t_c) = \frac{d^2}{dt_c^2}\Delta T(t_c)$.

- The **resolution** of a clock is the smallest interval by which the clock is updated. This value refers to the timescale of the clock itself instead of UTC. It is a lower bound on the uncertainty of the reported time; a low resolution of a clock is, however, no guarantee for its accuracy.
- Two clocks have a **relative offset** $\Delta T_{12}(t_c)$ which is the difference of their offsets to UTC, $\Delta T_{12}(t_c) = \Delta T_2(t_c) - \Delta T_1(t_c)$. For latency measurements, this value is more important than the plain offset.
- The **joint resolution** of two clocks is the sum of both resolutions; constituting a lower bound on the uncertainty of any interval calculated by differences of values from both clocks.
- **Synchronized** clocks are accurate with respect to one another, $\Delta T_2(t_c) = \Delta T_1(t_c)$. This aspect is very important for latency measurements, since the precise assessment of time intervals only depends on the relative offset of two clocks and their skew and drift; the absolute offset is, however, of no concern.

When the timestamping of data-units is performed by separate measurement probes which rely on different clocks, the following fundamental question arises: Do the clocks have to be synchronized, in order to enable One-Way Delay (OWD) measurements or is it possible to infer on OWD from one or multiple Round-Trip Time (RTT) measurements? In [53] it is shown, that in a connected network with an arbitrary amount of nodes N and a number of E directed links between them, it is only possible to obtain at most $E-(N-1)$ independent equations from any arbitrary combination of RTT measurements. Consequently, it is not possible to absolutely infer on the number of E OWDs. Supporting work has shown, however, that the restrictions on the individual OWDs imposed by the minimum propagation delay and minimum RTTs may lead to strong restrictions on the combination of OWDs for highly connected networks [54] [55]. Minimization strategies base on the *least-squared error* and *maximum-entropy* principles lead to acceptable OWD estimates in such a case. The proposed method has the disadvantage, that (i) the involved delays are assumed to be stable over time and (ii) fixed clock offsets are assumed for the involved nodes; in other words, all skews and drifts are zero. These assumptions requires high stability within the network and cannot be guaranteed in mobile cellular networks. It follows that for the measurements presented within this thesis, clock synchronization is required, in order to obtain the desired results.

3.1.1 Measurement Strategies

Obtaining synchronized clocks at multiple measurement points bears the difficulty that the timing information must be delivered to multiple locations. This information is sensitive on delay on its own; which has to be taken into account by designing the respective measurement setup. Several concepts may be deployed which allow for synchronized timestamping (but not necessarily involving synchronized clocks); they are summarized into three categories:

- **Single clock, single measurement point**: This most simple scheme supports only one measurement point. A single measurement probe timestamps observed packets at the respective location, which implies that all timestamps are provided by the same clock. In this case the clock does not need to be accurate, however, low skew and drift should be guaranteed compared to the delay values which are

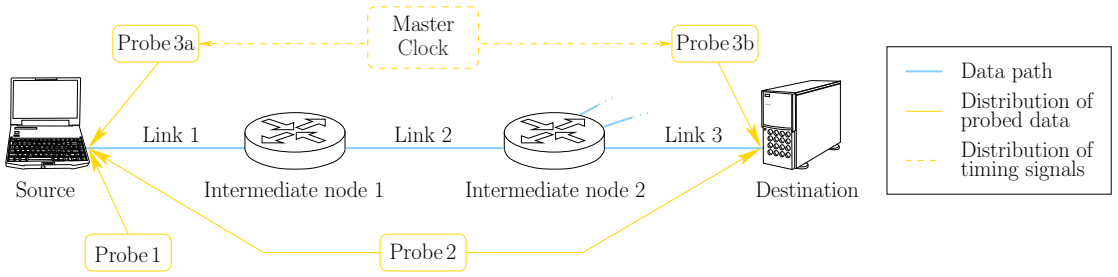


FIGURE 3.1: Delay measurement strategies with regard to timekeeping. Three scenarios are illustrated: (i) Probe 1 corresponds to the *single clock, single measurement point* scenario, (ii) Probe 2 to *single clock, multiple measurement points* and (iii) Probe 3a and 3b to *multiple clocks, multiple measurement points*.

assessed. This method is mostly deployed for RTT assessment, for example by the popular *ping* tool and RFC 2681 [17]. For further work deploying this strategy refer to [56] [57] [58] [59] [60]. A corresponding illustration is given in Figure 3.1, indicated by Probe 1. This measurement setup does not permit to measure OWD or any other delay on the route section of the data-units.

- **Single clock, multiple measurement points**: This scheme implies that *probed information* concerning the data-units is brought from multiple remote measurement points to a single timestamping device. It is represented by Probe 2 in Figure 3.1. Thereby, large distances may be bridged between the measurement point to the timestamping probe; causing the probed information to experience itself a delay, which is equivalent to an artificial clock offset. An accurate delay assessment is only possible if this artificial offset is known; which may be difficult to achieve for practical reasons. The only clock must not be accurate; however, low skew and drift should be guaranteed compared to the delay values which are assessed. This setup is most often deployed for a special arrangement of the links in the data route, namely, when links are remote entities on the data route (e.g., Link 1 and Link 3 in Figure 3.1) but coincide at a specific physical location. When the timestamping device is close to this location, the dedicated probing links are short and the artificial offset is negligible. Such a scenario is, for example, easy to arrange in cellular networks [61] [62] [63]. A wireless client and a cabled server can reside on the same physical machine, hence, deploy the same clock for timestamping. This measurement setup permits to measure OWD on each route section of the data-units; however, serving multiple measurement points at large distances is usually rather expensive in terms of hardware.

- **Multiple clocks, multiple measurement points**: This scheme requires multiple probes possibly placed at remote locations, directly at the links at which data-units shall be captured. Instead of distributing the probed information, timing information is distributed among the probes. Probes 3a and 3b given in Figure 3.1 are respective examples; both obtain timing information from a dedicated master clock. In this case the master clock must not be accurate; however, low skew and drift should be guaranteed compared to the delay values which are assessed. Further, synchronization mechanisms deployed at the timestamping probes should guarantee a good agreement of the respective clocks and the master clock. Since the advent of cheap and accurate methods for synchronization, the scheme is widely adopted for latency measurements [64] [65] [66] [67]. Measurement probes

with synchronized clocks have the advantage that any delay metric can be assessed with moderate hardware expenses.

The measurements presented within this work are exclusively performed by following this last strategy. For that reason an in-depth review of synchronization methodologies for multiple clocks is presented in the next section.

3.1.2 Probe Synchronization Techniques

Diverse synchronization strategies for multiple timestamping devices are outlined in literature, see [68] for a summary. Depending on the required accuracy of the respective delay measurements different strategies are recommended. If long term measurements are performed (e.g., delay variation over long periods, daily patterns), it is important to guarantee a clock frequency which is tightly coupled to UTC. When ultra short delays shall be assessed (e.g., OWD jitter, short high-speed links), then a low clock resolution and a tight synchronization at the measurement instant are of interest. Synchronization methods can be divided into two groups: (i) retrospective and (ii) real-time methods. Real-time methods allow for a bounded maximum clock offset; however, they mostly require additional hardware, which may be very costly if the maximum clock offset is low. Retrospective methods do not require additional hardware; they exploit the fact that usually multiple delay measurements are taken over long periods, which allow for a very accurate removal of any skew and drift, a constant yet small offset may however remain. Further, both synchronization paradigms are possibly combined, allowing for the removal of fluctuations of the synchronization control loop during measurement time. The following paragraphs give an overview of both strategies.

3.1.2.1 Retrospective Clock Correction

This strategy works with free running clocks during the measurement time and corrects the clock skew and drift in a post-processing step, i.e., during the calculation of the one-way delay. Nevertheless, if real-time synchronized clocks are in place, the respective synchronization performance can be improved using retrospective methods. The basic idea is to utilize the measured OWD values during the measurement period and infer on feasible values of the clock offset during this period. This problem is similar to the idea presented at the beginning of this section (cf. [53]), where RTT measurements are used to infer on OWD. The objective is however different, since the delay estimation according to [53] aims to infer on the clock offset, whereas the methods presented below aim to compensate the clock skew and drift.

Retrospective clock correction works very well when timing information can be exchanged on a fast and symmetric control link; while the targeted OWD measurements are simultaneously performed on a link with reasonably higher latency. An example is given in [69, pp. 93ff.], where mobile cellular OWD measurements are supported by the exchange of timing information on an additional wired link.

A survey on state-of-the-art correction algorithms is given in [70]. Early schemes focus on the removal of the clock skew only, a task which is equivalent to linear regression on the delay values. However, standard distance metrics are not applicable for this purpose, because the most valuable information about the clocks' synchronization states is carried by the minimum delay values. In [71] an algorithm is specified which accounts for this issue. Further [72] approaches the problem by formulating it as a linear programming algorithm.

A further prominent effect encountered are the so called *clock hiccups*. Accordingly, the clock jumps at specific points in time in order to compensate for time or frequency

offsets. This is especially the case if retrospective correction method are combined with assisted real-time synchronization. An algorithm proposed in [73] is able to handle this issue; thereby, the OWD time series is split into several intervals on which no hiccup events took place. On each of those intervals a separate skew compensation is performed by means of convex hulls. The weak point of this algorithm is the limited ability of detecting higher amounts of hiccups, which is tackled by supporting work [74] based on a clustering approach. Several other algorithms dealing with this issue are presented in literature, e.g., [75]. Correcting clock jumps by considering OWD yields an ambiguity between sudden changes in the clock and in the OWD itself. In [76] this issue is resolved by a fuzzy-based approach.
The complexity of the proposed algorithms may become prohibitively high, raising the need for simpler ones. The authors of [77] propose several simple algorithms and discuss the possibility of online skew removal (during the measurement period). Thereby sliding-window techniques are deployed. Summarizing, all the presented clock correction algorithms assume constant or piece-wise constant clock skew; they do not account for continuous clock drifts.

3.1.2.2 Assisted Real-time Clock Synchronization

The real-time synchronization strategies work on the principle to share timing information among clocks (possibly over large distances) and adjust the timestamping clocks accordingly, such that all clocks report the same time. This approach is by far the most popular for OWD measurements for the variety of available software and hardware, which strongly simplifies the arrangement of a measurement setup. It consists of two essential elements: (i) dedicated hardware for the distribution of timing information and (ii) a control mechanism which adjusts the clock (e.g., realized in software).
The task of the controller is to steer the clock based on the provided timing information. Abrupt changes and overshooting of the control loop must thereby be avoided. Further, the timing information is usually polluted by noise, which has to be eliminated by the controller. General introductions to timekeeping can be found in [78] [79], a review specific to network computers is given by [80]. Further, short and clear discussions of control loops for synchronization are provided in the context of the NTP protocol by [81] and the respective specification documents RFC 1305 [82] and RFC 5905 [52]. Software conforming to those specifications is available for most operational systems; the specified algorithms are therefore among the most prominent in practice. Newer algorithms [83] utilize additional counter registers provided by modern processors, which allow for an increased interpolation accuracy for high resolution timestamps. Appendix C provides an overview of the timekeeping hardware and software I designed within the scope of this thesis and gives a respective performance evaluation.
The distribution of timing information is possible by diverse techniques; the respective choice shall be based on a trade-off between accuracy and cost. Driving factors are: (i) the clock source, (ii) the distance to be bridged, (iii) dedicated hardware support and (iv) error sources due to delay variations on the timing link. Overviews of setups and comparisons are given in [84] [85] [68], the most prominent approaches are:

- **Network Time Protocol (NTP)** is the most popular method for clock synchronization. It enables the distribution of pure UTC timing information over the Internet and is based on the User Datagram Protocol (UDP) protocol (port 123). The specification of NTP are made by the Internet Engineering Task Force (IETF) in the documents RFC 1305 [82] (NTP-v3) and RFC 5905 [52] (NTP-v4). NTP uses

a hierarchical clock organization, named *clock strata*, with *Stratum 0* clock as references (e.g., atomic standards, GPS clocks). The achievable accuracy is strongly dependent on the connection to the Internet. Tens of milliseconds are possible over public wired links, whereas 1 ms of synchronization accuracy can be achieved over local networks. It is feasible to deploy NTP for accurate delay measurements; however, the timing information must be delivered by a separate and more reliable link than the link under test.

- **Global Positioning System (GPS)** allows for accurate localization on the earth's surface. This is achieved by the distribution of accurate timing information over satellites, which carry atomic clocks. Accordingly, GPS time can serve to synchronize commodity clocks; whereby they inherit the high precision and stability of atomic standards. Accuracies of down to some hundreds of nanoseconds can be achieved by this approach, provided that dedicated hardware is deployed [86] [87]. By the synchronization of commodity desktop PCs an accuracy of roughly 10 μs is achievable [6], cf. Appendix C. GPS receivers require a clear view to the sky in order to work properly; being disadvantageous in specific scenarios.

- **Precision Time Protocol (PTP)** is a protocol to synchronize clocks over a network also know as IEEE 1588 [88]. Unlike NTP, PTP is designed for the use in industrial-automation environments with short symmetric links and provides several possibilities for improved synchronization: (i) enhanced OWD estimation and compensation, (ii) hardware packet timestamping support and (iii) selection of a reliable time source by a *best master clock* algorithm. According to [89] IEEE 1588 is designed to fill the niche between GPS and NTP. The achievable accuracy is in the sub microsecond range for dedicated hardware. Implementations for desktop PCs are available [90] [91].

- **Daisy-chain of Pulse Per Second (PPS) signals**: For OWD measurements accurate synchronization to UTC may not be of prior concern. Instead only a tight synchronization of multiple timestamping clocks is required. In such cases timing information provided by one clock can be delivered to other involved devices by a simple electrical impulse; often named PPS since it is usually triggered once per second. This way of distributing timing information is cost-efficient and provides good synchronization performance [92], comparable to GPS and PTP. It can be either deployed with special hardware [87] but also with ordinary desktop PCs [93].

- **White Rabbit** is a project based on the combination of Synchronous Ethernet (SyncE) [94] and PTP. In combination with optical fiber connections on the physical layer a sub-nanosecond synchronization accuracy is reported [95] for several kilometers of distance.

3.2 Packet Injection and Capturing

Latency measurements are performed for diverse reasons, resulting in a wealth of different measurement setups. They all have in common that data-units are injected on a route section of the network and captured (recorded and timestamped) at its beginning and end, respectively. In the following I discuss the most common approaches for the injection and capturing of packets.

3.2.1 Traffic Sources: Active and Passive Measurements

An important aspect of delay measurements not discussed so far, is the origin of the data-units which experience latency. This issue can be regarded as most distinctive feature of network measurements and is tightly related to the respective problem statement. In general two traffic sources can be discerned and, accordingly, network measurements are split into two categories:

- **Active probing/measurements:** The researcher performing the network measurement injects actively packets and monitors the corresponding response of the network.
- **Passive probing/measurements:** The researcher relies on the ordinary traffic (cross-traffic) available in the network (produced by real users) and passively monitors the network response.

The problem statement has immediate consequences on the choice of the source of the network traffic. If properties of the network shall be assessed, then the preferred approach is active probing. This way the researcher can directly control the stimulus (probing packets) and trigger a statistically significant amount of reactions of the network (note that in case of strongly intrusive probing traffic the passive approach may be preferable). On the other hand, when properties of the cross-traffic (background traffic) shall be assessed, then it is more convenient to deploy passive measurement strategies (note that it is also possible to perform such measurements actively, cf. Section 2.3.1.1 and references therein).
Both measurement paradigms offer distinct controllability and observability of parameters during a measurement. Typical problem statements for *active* measurements are solvable with **black-box tests**. They are concerned with the overall network performance and often only require the free choice of the probing pattern exchanged between client and server. An exact understanding of the underlying procedures of the network is of secondary interest. Further, the cross-traffic is not of primary concern and treated (e.g., suppressed) by statistical methods. Consequently, observability of external parameters is not required. Typical problem statements for *passive* measurements are solvable with **white-box tests**. These include (i) traffic analysis, (ii) network optimization and (iii) anomaly detection. White-box tests require a deep understanding of the network and the underlying procedures, the fine-grained observability of several external parameters are of major concern. It is especially important, that the measurements are non-intrusive, what can only be achieved by dedicated measurement hardware at nodes or links. Consequently, controllability of the traffic patterns is not desired.
Related literature providing overviews and guidelines for the selection of one of the mentioned measurements paradigms is available at [68] [96] and [97, pp. 211ff.]. Further, passive and active measurements are not competing strategies. There exist specific measurement setups which can be accounted to both groups. Such setups are referred to as *hybrid*, see [4] for a respective treatment.
Secondary considerations often influence the choice of the traffic sources; thus, the deployment of the respective probing paradigm. Such considerations are (i) legal and ethical issues and (ii) hardware considerations. They are approached in the following sections.

3.2.2 Legal and Ethical Considerations

Legal issues in the context of network measurements are often neglected or overlooked; however, especially passive measurements (concerned with private data of real users) are

subject to these, since the explicit consent of users for the deployment of their data for scientific analyses is rarely given. Several countries released regulations concerning the treatment of private user data. Such regulations do typically *not* consider the recording and evaluation of network traffic for research purposes; thus, cause Internet research to reside in a legal gray area. This issue is discussed in [98] [99], which give an overview of current regulations and provide advices for academic researchers.
The current scientific practice is to obfuscate sensitive data to fulfill privacy requirements for both users and service providers. Prevalent approaches are thereby: (i) **anonymize** sensitive information, (ii) **remove** sensitive information and (iii) store raw data only **temporarily**. Those regulations are often formulated in policies (between researchers and data providers); which however seldom incorporate specific information on how to anonymize, what data to remove or how long data can be retained.
The optimum amount anonymization is strongly related to the problem statement. Approaches reach from: (i) hashing of addresses, yielding anonymized addresses without one-to-one mapping to the real address space, (ii) translating addresses to pseudonyms, leading to a one-to-one mapping and (iii) prefix-preserving anonymization, which is maintaining information about the network and enables a one-to-one mapping. The last method is thereby prevalent for network measurements, cf. [100]. Concerning latency measurements, the last two methods are applicable, since the pseudonyms are unique due to the one-to-one mapping. This allows for a correlation of several traces captured by probes at different locations in the network. However, all probes have to deploy the same mapping in order to allow for the merging of traces.
Removal of sensitive information is usually performed by *payload truncation*. Accordingly, only the first part of each packet is recorded (containing the protocol headers, cf. Appendix A), the tail of the packet is discarded. This procedure comes with the additional benefit that the amount of data to be stored is reduced. However, the boundary between headers and payload is rather fuzzy. The content of the headers above the transport layer (cf. Section 2.1) depends on the higher protocols, and may contain both sensitive (e.g., HTTP, DNS) and insensitive information (e.g., SSH). It is common practice to truncate all packets after a specified number of bytes, which however raise privacy concerns. A further ambiguity is constituted by malformed packet headers. Dropping such packets yields a biased view on the network traffic, especially since malicious traffic is leveraging this kind of packets.
The **sharing** of measurement traces is indispensable for the facilitation, reproducibility and verifiability of research. The network research community has made tremendous progress in this regard within the last few years [101]. However, sharing of traces is especially delicate with regard to legal issues. Several studies [102] [103] showed that it is possible to infer on sensitive information from publicly available data sets. Thereby, information about users (e.g., host behavior) as well as the network itself (e.g., topology) have been disclosed. Ongoing work is concerned with better privacy conservation [68]; proposals are (i) a hybrid policy and technology framework enforcing privacy obligations, (ii) a *"move the code to the data"* framework, relying on the publisher of the measurement data to perform analysis for third parties and (iii) a secure query language enabled to instantaneously provide anonymized statistics of sensitive data sets.

3.2.3 Hardware Requirements

Depending on the measurement strategy, different kinds of hardware are required. Active measurements require data sources (clients) and data sinks (servers) but, typically, no capturing devices residing on intermediate links. Passive measurements, on the other

hand, are based on traffic capturing devices which are installed on one or more specific hops. Devices at the source or sink are unnecessary.
Data sinks and sources for active measurements are mostly commodity hardware, unless specific border cases shall be tested. Commonly those two devices further act as sensing probes, which is feasible when end-to-end performance metrics are under investigation. The measurement setups are often very cost efficient. Performing massive amounts of measurements is, on the other hand, less attractive since: (i) transmission of high amounts of data over public networks is expensive, (ii) intrusive measurements exceeding certain thresholds are inconvenient for providers and (iii) deployment of a massive amount of measurement probes is costly in terms of money and time. In recent years **crowd-sourced** measurement approaches became popular [104] [58] [105], which manage to overcome the drawbacks mentioned above. In such scenarios private users contribute to the measurement by hosting the measurement software on their devices and bearing the costs for the transmission of probing traffic.
Passive probing relies on capturing devices within the network. These can further be divided into **software-based** capturing (program installed on a node) and **hardware-based** capturing (packet sniffer on a link). Software based capturing tools are inexpensive and available for most network nodes (e.g., [106]). However, they suffer for two drawbacks: First, software based tools consume processing power and storage, which could lead to a performance degradation of the network nodes in overload scenarios. Second, reporting protocols, interfaces and data structures are rarely standardized. An example for such a standard is IP Flow Information Export (IPFIX) [107] for the reporting of flow related information. Hardware based tools capture data traffic directly on the link, which requires dedicated hardware in place. An inexpensive solution is to activate port mirroring on existing hardware (e.g., routers, switches), and place a commodity device on the mirrored port to record the traffic. State of the art devices support high capture rates; however, with reduced timestamping accuracy [108] [109]. Mirroring is supported by most high-end products out of the box, the drawback is however the increased processing load. The alternative is to install wiretaps on the link under observation. Wiretaps are available for copper and optical fiber, with supported data rates up to 10 Gb/s [110] [87] [111] [112]. This solution overcomes the disadvantages of port mirroring, namely, it is strictly non-intrusive; the setup is however costly and the installation causes service interruptions.
In order to combine (correlate) captured packets from multiple interfaces (e.g., for calculating the latency), they have to be stored and transfered to a central processing unit. Unfortunately, there is no standardized format to be deployed for this task; several formats have been evolved in parallel [113]. A very common standard is the *pcap* format [114], which is supported by most software suits. Conversion between different standards is possible, however, may yield loss of information (e.g., timestamp precision).
The amounts of data to be traced may reach dramatic scales, especially for the passive monitoring of high-speed links. In such cases the recording of data is non-trivial, since captured packets have to be buffered for a significant amount of time before respective processing can take place. The bottleneck is thereby constituted by the storage capacity, yielding the retention of complete packet-level traces infeasible. In such situations several solutions are proposed:

- **Filtering**: Packet headers are inspected during the capturing process. If a packet does not meet certain pre-defined criteria it is discarded. Respective approaches are discussed in [115].

- **Sampling**: Similar to filtering, this approach discards packets. This is however done on the basis of an external process instead of properties of the packet itself [115] [116]. The advantage is thereby that systematic errors are largely avoided, especially for measurements with general purpose.
- **Payload truncation**: Only packet headers are stored, the rest of the packet is discarded. Truncation can be performed after a fixed size of bytes or on the basis of the involved protocols. The advantage with regard to latency measurements is that no information is lost. Further, privacy constraints are a priori satisfied Section 3.2.2.
- **Aggregation**: This approach aggregates captured packets (e.g., flows, sessions) and records statistics about such clusters. The amount of data to be stored is hence dramatically reduced. This approach is often encountered in software-based packet capturing (e.g., IPFIX [107]). Latency measurements on the basis of aggregated traffic are subject to current research [117].

3.3 Measurement Design

The final step required to perform latency measurements is to define a measurement methodology, in other words to design the experiment. Thereby a framework is created which describes the complete experiment; e.g., the network under test, the hardware setup and the evaluation procedure. The explicit establishment and documentation of the measurement design is an invaluable tool for troubleshooting and the reproducibility of the measurement.
The common steps, loosely based on [118] [119] [4], in a practical experimental environment are to:

- Formulate the problem statement,
- Select the metric,
- Design the measurement gauge (hardware setup),
- Identify and classify influencing factors,
- Define the measurement procedure,
- Perform the experiment,
- Evaluate the data, and
- Interpret the results.

Usually these steps are part of an iterative process, where it can be necessary to start over and improve the problem statement. Possible reasons are, for example: (i) the initial assumptions (models) are found to be violated or (ii) intermediate findings yield the measurement design suboptimal. In the following the above steps are discussed in detail, including examples taken from the measurement study presented in Section 2.3.2.3.

3.3.1 Formulation of the Problem Statement

Measurement studies are often chicken-and-egg problems: A model is the basis for a measurement study, which in turn, yields a refined model. The first step in the measurement study is thus to *assume a model* and *formulate a problem statement* based on it. A common beginner's mistake is thereby to ask the wrong questions; a measurement study provides *no* answer to questions such as **"How does the network behave?"**, instead it can answer **"Does the network behave according to model XY?"**
Besides of deciding on a preliminary model, this very first step shall clearly express the goal of the measurement process. One must understand the reason for the measurement campaign and already define which outcome would be useful. It is especially important to realize the required amount of generality of the results. For example, the designer should understand if he or she is searching for the delay of one specific packet of a known application passing an arbitrary network at runtime or if the goal is to investigate the delay behavior of a whole class of networks. Furthermore, this is the step which roughly determines the overall duration of the measurement, the desired sample size and the precision of the result.

> *Example*: In the measurement study presented in Section 2.3.2.3, I intend to show that modern High Speed Packet Access (HSPA) networks can be reactive. The preliminary model is thereby given in Definition 2.5. The goal of the measurement campaign is to provide evidence that public networks *can* be reactive; hence, it is enough to show that Definition 2.5 holds for one specific HSPA network. Additionally, Eq. (2.9) and Eq. (2.10) shall be disproved for the network under investigation.

3.3.2 Selection of a Metric

The exact *metric* or *response variable* of the experiment is defined in this step. The definition of delay provided here must be as precise as possible; it heavily influences the overall experiment. This step shall cover all parameters with immediate effects on latency, as discussed in Section 2.2; furthermore, parameters related to traffic sources, cf. Section 3.2.1, shall also be defined. A summary of parameters to be fixed is given in the following checklist:

- The data-unit and the conditions for the respective validity (cf. Section 2.2.2).
- The measurement points in the network (cf. Section 2.2.3).
- The delay metric (cf. Section 2.2.4).
- The traffic source which injects the data-units (cf. Section 3.2.1).

Those parameters lay the foundation for the measurement gauge. In contrast to the problem statement, this step already includes secondary considerations such as estimated costs and data accessibility.

> *Example*: In the measurement study presented in Section 2.3.2.3, the delay metric is defined as in Section 2.2.4. Further, the interfaces directly at the client and the server were chosen as measurement points, yielding pure end-to-end latency. The delay metric was specified according to Definition 2.2. Finally, as traffic source active measurements with one server and one client have been chosen, where only uplink traffic is considered.

3.3.3 Measurement Gauge Design

The required measurement tools (hardware or software) can be designed based on the definitions provided in the previous step. The selection of the measurement approach depends on expenses, availability, reliability, accessibility, complexity and precision. This process profits from an initial cost-benefit analysis based on prior steps.
The most important choice concerns the timekeeping methodology, confer Section 3.1. Further, the deployment of hardware-based or software-based capturing tools is determined, see Section 3.2.3. The capture devices produce timestamped tickets for each packet; the format of these tickets must thus be stated. This includes legal and ethical aspects (e.g., anonymization) as mentioned in Section 3.2.2. After the capturing process, the tickets have to be transfered to a common location and combined in a post-processing step. This can either be done online (e.g., as performed by the *ping* tool) or offline (e.g., as for the measurements presented throughout this thesis). Dedicated software is usually the method of choice to perform this task. Handling of exceptions (e.g., missing or malformed packets) must be implemented according to the definition of valid data-units (cf. previous step) and a format for meta-data must be specified (e.g., log-files).

> *Example*: In the measurement study presented in Section 2.3.2.3, the packet capturing has been performed deploying the software-based approach. The timekeeping was handled by GPS synchronization of server and client. Captured packets have been stored on both machines in the form of a *pcap* traces. The combination of tickets to delay values was performed offline by a custom script written in *Perl*; whereas meta-data is provided in plain text format.

3.3.4 Identification of Influencing Factors

Latency is a variable that comprises contributions of multiple components along the data route. Several parameters influence latency, especially when the measurement points are separated by many hops. The comprehensive identification of those parameters is crucial for a sane measurement study. A list of possible influences is given in Table 2.3, which is tailored to mobile cellular networks. The parameters concern several aspects reaching from the traffic pattern over cross-traffic to the measurement gauge itself. It is advantageous to assess the dependence between parameters to simplify the design of the measurement procedure. Thereby, it is important to distinguish between causation and correlation, in order to not confuse **causes and effects**.
After the identification of the influencing parameters they have to be classified. First, each variable has to be assigned to one of the two groups: (i) **design variable** or (ii) **nuisance variable**. Design variables are those of which the measurement study shall provide a statement; usually they are already included in the preliminary model formulated in the problem statement (first step, Section 3.3.1). Nuisance variables are not of major concern for the measurement study, but may affect the overall behavior of the response variable; hence, must be considered during the design and evaluation phase.
Second, a classification in terms of accessibility is required. It must be determined if the variables are: (i) **controllable**, (ii) **measurable** or (iii) **hidden**. Controllable variables are factors which can actively be manipulated and measured during the experiment. Measurable parameters cannot be actively controlled but measured. Hidden variables can neither be controlled nor measured but are expected to influence the overall study (note, that design parameters shall not be hidden).

Example: In the measurement study presented in Section 2.3.2.3, the parameters presented in Table 2.3 are identified to influence the response variable (end-to-end OWD). Thereby, only the packet sizes π and packet inter-arrival times τ are classified as controllable design variables. Global and instantaneous data-rates, traffic on the reverse link, protocol related settings, hardware and provider are controllable nuisance parameters. The time of day is a measurable nuisance parameter; whereas all residual parameters are treated as hidden nuisance parameters.

3.3.5 Definition of Measurement Procedure

With the aid of a list of influencing parameters and the respective classification, this step is straight forward. Excellent discussions on this matter can be found in [118] [119]. This step shall ensure that:

- The influence of nuisance parameters on the experiment is suppressed. This can be achieved, for example, by randomization or blocking.
- Effects caused by design parameters must be distinguishable. It is achieved, e.g., by paired or factorial designs.
- The statistical significance of the samples must be guaranteed. The number of samples has to be defined together with the corresponding range.

This step comprises to trade off accuracy and generality against costs and temporal efforts. It is important to keep the objectives for the measurement study in mind, e.g., shall effects be verified, disproved, monitored or quantified.

Example: In the measurement study presented in Section 2.3.2.3, the traffic patterns have been designed according to the scenarios presented in Figure 2.7 (a). Several factors have been fixed, e.g., (i) the traffic patterns on the uplink and downlink are statistically equivalent and (ii) the deployed transport layer protocol is UDP. In order to cope with variations over the time of day (global workload) the single experiments (covering one scenario) have been interleaved and repeated over 24 h.

3.3.6 The Experiment

The experimental procedure should be automated to a large extend, in order to avoid human errors in this complicated procedure. Further, it is of huge benefit for traceability; e.g., to verify if unexpected effects occur due to a faulty measurement setup. Additionally, the reproducibility of the experiment is guaranteed; being advantageous when measurements are performed by iteratively improving the respective design.
During the measurement the experimenter has to continuously monitor the procedure. Errors during the experiment usually yield bad samples and require to start all over. The timely detection of such errors saves both time and money. It is recommended to perform short trial runs of the whole experimental procedure. They provide insights into the consistency of the measurement results and enable the verification of the measurement setup.

Example: In the measurement study presented in Section 2.3.2.3, the experiment has been performed in a fully automated fashion. Client and server are connected via a primary connection (link under investigation) and, additionally, via a secondary wired Ethernet connection on which all control-information is exchanged.

When the client is started, it starts the sever, initiates time synchronization at both sides, transfers the downlink traffic pattern and resets the modem. Then the experiment is started. After the experiment is finished, the measurement trace is transfered to the client, log-files are evaluated and the traces are merged.

3.3.7 Evaluation of Data

In order to obtain objective results, statistical methods have to be deployed for the evaluation of measurement data. Software tools for this task are largely available. Since the amount of influencing factors (dimensions) of latency is big (cf. Table 2.3), one cannot expect to obtain a comprehensive set of measurement data (this would correspond to a multidimensional distribution function). Accordingly, the statistical analysis performed in this step cannot compensate for misconceptions in preceding steps.
Basic tools for the evaluation of data are **simple statistics**, such as average, variance and correlations. They should be complemented by **graphical methods** (e.g., histograms, scatter-plots for the visualization of correlations). Those methods are often sufficient for the evaluation of measurement data. In case the problem statement requires the application of more specific methods (e.g., parameter estimation for specific models), basic tools are still valuable for sanity checks and should never be omitted (e.g., verification of the validity of the involved models). An invaluable additional information on estimated parameters are **confidence intervals**, which can be obtained, for example, by resampling methods [120]. Many problems can further be cast into a **hypothesis-testing** framework [118] [119]. In other cases it is more adequate to build an **empirical model**, which mirrors the relation between the response parameter and the design parameters (this facilitates the dissemination of the results).

Example: In the measurement study presented in Section 2.3.2.3, the goal was to provide evidence that HSPA networks are reactive according to Definition 2.5. To show this, I conditioned the measurement values obtained for different scenarios on the checkpoints introduced in Figure 2.7 (a). Accordingly, the hypothesis that *all scenarios conditioned on the same checkpoint exhibit the same latency figure*, needs to be disproved. This is shown in Figure 2.8 (a), where several scenarios exhibit obvious deviations for Checkpoint 4–7. The enormous sample size (20 000 packets per scenario) leads to the rejection of the hypothesis with very high probability.

3.3.8 Interpretation and Recommendations

This last step is required to draw final conclusions from the measurement study. Since measurement studies are often **iterative** procedures, it has to be decided upon further actions. The questions to be answered are: (i) **"Are the results satisfactory?"**, if not, (ii) **"How much can I gain from another measurement run?"** and (iii) **"Is this worth the effort?"**. The results obtained in the present measurement run can be used to refine the model required in the problem statement phase (cf. Section 3.3.1) and to modify the list of influencing factors (cf. Section 3.3.4). As a rule of thumb, no more than 25% of the resources shall be devoted to the first measurement run [119, p. 19].

Example: The measurement study presented in Section 2.3.2.3 was repeated several times. Effectively, Definition 2.5 (constituting the central part of the problem statement) as well as Table 2.3 (list of influencing parameters) were altered several times before they attained the present appearance.

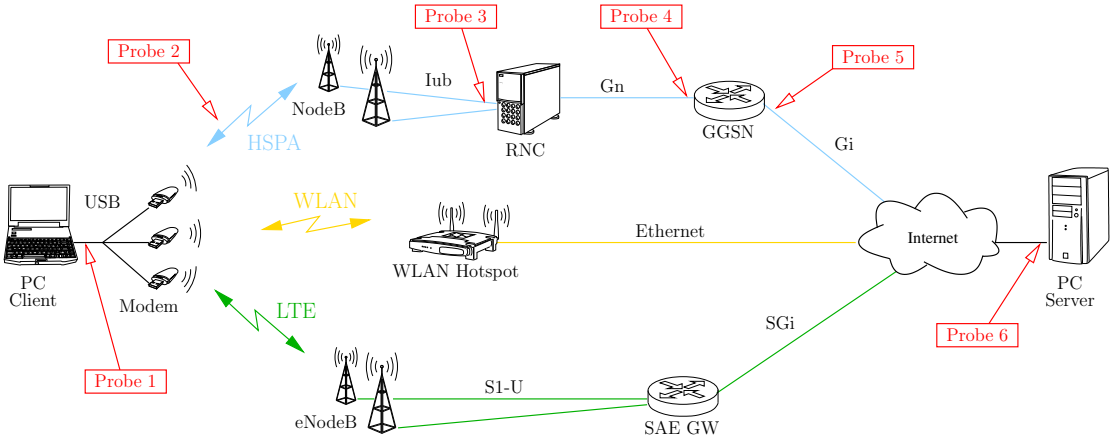


FIGURE 3.2: All measurement setups deployed throughout this thesis.

3.4 The Applied Measurement Setup

Several measurement studies are presented throughout this thesis. They mainly concern wireless networks and deploy similar measurement setups. In this section I summarize the deployed strategies into a single framework.

The setup comprises three wireless communication technologies: (i) Wireless Local Area Network (WLAN), (ii) Long Term Evolution (LTE) and (iii) HSPA. A respective illustration is given in Figure 3.2. Thereby the definition of the delay metric is according to Section 2.2, with measurement points indicated in the figure. Specific studies may thereby only require a subset of the outlined setup; e.g., the measurement study presented in Section 2.3.2.3 only considers the uplink end-to-end OWD in an HSPA network; hence, packets are only transmitted over the top-most branch and captured at Probe 1 and Probe 6.

The **hardware** used for packet capturing consists of software-based probes (i.e., Probe 1 and 6) and hardware-based probes (i.e., all others). Respective details are provided below. The capturing format depends on the probe, whereas (i) Probes 1 and 6 produce *pcap* [114] trace files, (ii) Probes 3–5 produce *erf* [87] traces and (iii) Probe 2 produces traces in a custom format. The respective combination and the calculation of the delay values is usually performed **offline** on the client.

The **timekeeping** is performed with GPS synchronization at all involved interfaces. Thereby three different hardware setups have been deployed, outlined in detail below. The estimated accuracies are statistically (offline) evaluated and verified to be below 25 μs (worst case) [1] [6] [87]; this is satisfactory for the targeted range of delay values (0.5 ms to 5 s). Measurements presented in the present work are performed without retrospective clock correction. However, online sanity checks and accuracy estimations are performed during all measurements by evaluating variations of the arrival instants of the synchronization pulses. The thereby estimated clock offset must be below the threshold value $\Delta T_{\max}$=50 μs in order to yield a valid measurement. Otherwise, the entire measurement run is discarded and repeated.

The setup outlined in Figure 3.2 suggests an **active** delay assessment procedure. This is however not required, since the probes installed at intermediate interfaces of the HSPA network allow for the **passive** observation of ordinary users. Nevertheless, all delay measurements presented within this thesis (e.g., Section 2.3.2.3, Section 4.1.4, Section 4.2 and Appendix D), as well as the traffic traces presented in Section 5.1 are obtained by active measurements. Only the measurements presented in Section 5.3 are based on passive

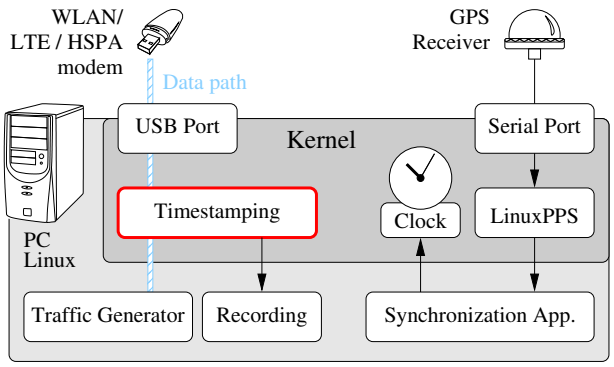


FIGURE 3.3: Measurement probe for packet capturing at source and destination hosts.

measurements. Anonymization and payload-truncation has been performed, adhering to the considerations presented in Section 3.2.2 (cf. Section 5.3 for more details).

3.4.1 Probe 1 and 6: Source and Destination

The measurement probes, capturing packets at the interfaces at source and destination, are software based probes, residing at the respective end-hosts. This is depicted in Figure 3.3. The advantage of this approach is that it is completely hardware and network independent; i.e., changing the communication technology only requires to connect a different modem to the PC.
The packet capturing and timestamping is thereby performed by the *tshark* application [114], which saves the packet traces in a local file. Since the timestamping operation is performed directly in the kernel domain, the uncertainty caused by this operation is assumed negligible. The PC has to be synchronized to UTC in order to produce timestamps which are comparable to those of other probes. This is achieved by GPS synchronization, deploying: (i) a *Sparkfun EM-406A* GPS receiver with PPS output, (ii) the *LinuxPPS* driver [93] for capturing the PPS signal at the serial port and (iii) a custom synchronization application (cf. Appendix C and [6] [121]) which I designed for ultra-fast convergence to UTC. The achievable accuracy is strongly dependent on the hardware; it has been verified that values less than 10 μs are achievable on modern PCs. A possible drawback of this approach is that the traffic capturing process may be affected by the traffic generating process and vice versa; which would lead to a tremendous performance decrease. However, the deployed end-hosts are modern high-performance computers and the average rate of the generated data traffic is low (always below 200 kB/s). Accordingly, the overall CPU load never exceeded the 20%-level during measurements, which I consider sufficient for assuming no mutual influence between the two tasks.

3.4.2 Probe 2: The Air Interface

The measurement probe capturing packets at the HSPA air interface (Probe 2) is split into two devices: (i) a hardware-based probe for the uplink, capturing packets between the modem and the wireless link and (ii) a software-based probe for the downlink, capturing packets between the Base Station (NodeB) and the wireless link. Although the measurement points do not coincide on the link, I assume the delay between them as negligible. This is due to the low propagation time on the wireless link caused by the physical vicinity of User Equipment (UE) and NodeB (300 m distance causes 1 μs delay).

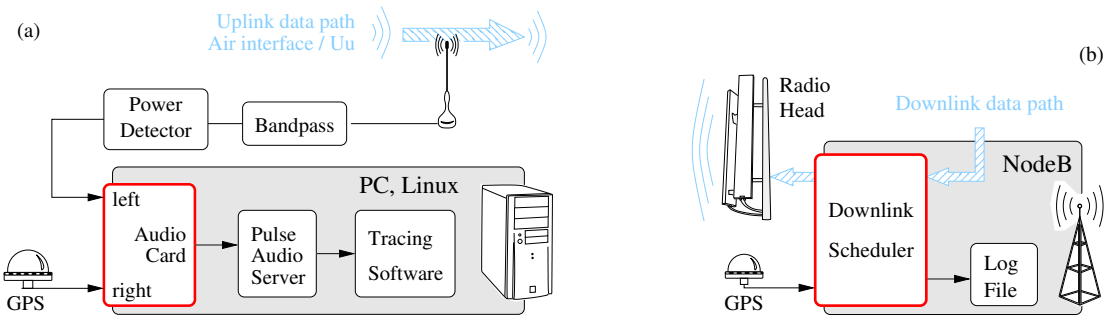


FIGURE 3.4: (a): Measurement probe for the uplink air interface. (b): Measurement probe for the downlink air interface.

The packet detection device for the uplink direction does not perform real packet capturing (i.e., detecting, decoding and recording of packets), but follows a different approach. The device directly identifies the start and end time of single packets by monitoring the transmission power of the UE. In High Speed Uplink Packet Access (HSUPA) an extra amount of transmission power is allocated to the UE via the Relative Grant Channel (RGCH), in order to transmit data in uplink on the E-DCH channel [22]. As long as the Inter Packet-Arrival Time (IAT) is long enough, this allocation step is performed for each packet; yielding an observable change in transmit power. Thereby the minimum required IAT depends on the packet size (e.g., roughly 100 ms for 1.5 kB payload size). The measurement device deployed for this task is depicted in Figure 3.4 (a). It consists of: (i) an antenna placed nearby the modem and a bandpass filter for the Universal Mobile Telecommunication System (UMTS) uplink frequencies (1920–1980 MHz), (ii) a Radio Frequency (RF) power detector circuit [122] and (iii) a PC with audio card, where the signal from the power detector is fed to the left channel, the PPS signal from the GPS receiver to the right channel. The decoding procedure requires to combine the traces from Probe 1–3 in order to correctly identify packets. It is outlined in detail in [1]. The timestamping accuracy is in the order of 25 μs for a 44.1 kHz audio card. This approach is, to the best of my knowledge, novel and was first presented in [1].
In downlink direction, the packet timestamping is achieved by consulting the logging information of the downlink scheduler at the NodeBs. The respective setup is depicted in Figure 3.4 (b). This information is aligned to UTC since NodeBs are required to be synchronized to common time slots. Also in this case only timestamps are recorded, without the information of the corresponding packet. Consequently, the decoding procedure again requires to combine the traces from Probe 1–3. Further, since the logging of scheduling information is not possible for every NodeB, this step has to be performed offline. Fortunately, the decoding and processing delay at the UE appeared to be deterministic, such that it is possible to infer on the timestamp at the air interface from the timestamp at the USB interface by a post-processing step. The timestamping accuracy is estimated to be better than 2.5 μs, since a corresponding synchronization accuracy of the transmit signals is required for the proper functioning of Wide-band Code Division Multiple Access (WCDMA) networks in TDD mode [123].

3.4.3 Probe 3–5: The Mobile Core Network

The tracing of packets exchanged at interfaces within the network of the mobile operator was performed with dedicated measurement equipment, developed within the *METAWIN* project [124] [96] [125] [126]. All links are traced by measurement probes which deploy GPS synchronized data acquisition cards [87]. Figure 3.5 shows the setup.

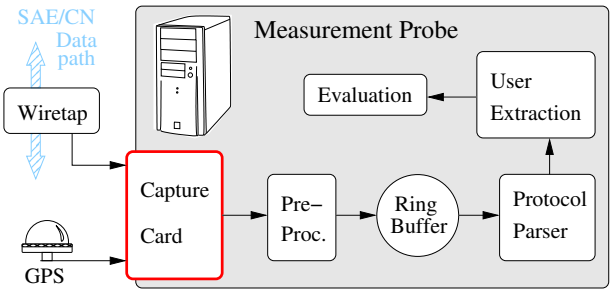


FIGURE 3.5: Measurement probes within the mobile network.

Accordingly, this capturing approach is hardware-based and does not impose any additional load on the network infrastructure. The timestamping accuracy is specified by the manufacturer to be better than 200 ns [86, pp. 97–98]. Due to the high data aggregation at the traced interfaces, the main challenge for the tracing devices consists of the extraction of the packets dedicated to the user of interest. This requires measurement nodes with high-speed protocol parsing capabilities and the possibility to correlate information from different protocol layers. The differences between the three deployed probes are mainly in software; each of the assessed interfaces inherits a different protocol stack; hence, each monitoring probe needs respective parsers.

3.4.4 Criticism

The presented measurement setup is of course subject to a trade-off between cost and accuracy. Thereby it was possible to realize the measurement probes at the end-nodes, Probes 1 and 6 (cf. Section 3.4.1), as low-cost probes and simultaneously achieve high accuracy. This was feasible due to the relative low data rates which are exchanged at the respective interfaces. However, installing measurement devices similar to Probe 3–5 at the place of Probes 1 and 6 would further improve the quality of the results.
The probes installed at the core-network, Probe 3–5 (cf. Section 3.4.3), the respective installation and the deployment entailed very high costs. They were only affordable due to the collaboration with other projects [124]. Consequently, I gained access to high-quality measurement probes for relatively low costs, however, only for limited time.
The measurement probe with lowest accuracy is Probe 2 (cf. Section 3.4.2). The reason is that the accurate timestamping of packets captured at the air-interface is an unconventional problem statement. Commercial hardware is not available; instead, it has to be designed from scratch. As mentioned in Section 3.4.2 several workarounds have been deployed in this context, in order to keep the design relatively cheap. The probe would benefit from specialized hardware for (i) a higher timestamping resolution and (ii) proper receiving and decoding of the radio signals. However, the respective development was regarded as too time and cost intensive for the present thesis.

Chapter 4

Benchmarking Wireless Networks

Equipped with the tools presented in the previous chapters, latency can be assessed for arbitrary packet networks. This chapter exhibits the real delay behavior of wireless networks in great detail and highlights encountered difficulties by performing the respective measurements. Further, it provides guidance for the interpretation of other measurement studies on delay in reactive and non-reactive networks.
Due to the reactiveness of mobile cellular networks which has been evidenced in Section 2.3.2, each sophisticated investigation shall incorporate considerations on traffic patterns, as presented in Section 4.1. With the right pattern(s) at hand the practitioner can proceed to evaluate the network behavior and extract significant delay benchmarks. The study presented in Section 4.2 shows such benchmarks for several settings within a High Speed Packet Access (HSPA) network; allowing for a clear ranking among them.

4.1 Dissecting Influences of Probing Patterns on Delay

The challenge for latency assessments in wireless mobile networks is to come up with a sane problem statement (cf. Section 3.3), which inherits a preliminary model with the capability to account for the reactiveness of the network. If this is not the case, a specific probing pattern may cause delay effects which are mistakenly attributed to the network in general.
Section 2.3.2 shows, that conventional requirements on probing traffic (e.g., non-intrusiveness, Poisson Arrivals See Time Averages (PASTA) property) are not sufficient for a comprehensive latency assessment in reactive networks, since:

- The latency in a reactive network depends according to Definition 2.5 on parameters of the probing pattern θ_p (including its history). Any measurement of the delay is conditioned on the respective probing traffic, yielding $\delta[n]{=}f(\theta_p, \psi, \phi)$; however, $\delta[n]{=}f(\theta, \psi, \phi)$ is usually the function of interest, considering $\theta{\in}\Theta$ as an unknown parameter.

- Specific probing patterns cover specific aspects (parameters) of Θ; however, there is no *perfect* probing pattern covering all aspects to a significant extend. Each probing pattern has its strengths and weaknesses; hence, relying on one single probing pattern entails the risk of undiscovered effects.

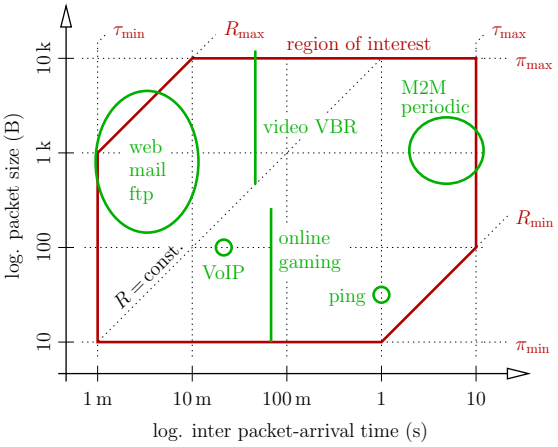


FIGURE 4.1: PS-IAT plane as a subset of Θ, including popular applications (green) and a more extended region of interest (red).

4.1.1 Related Work

Latency measurements received reasonable attention in literature. The measurement methodology established in this section focuses on the design of probing patterns for accurate and unbiased delay measurements. Related literature has already been introduced in the context of Palm calculus Section 2.3.1.4; it mainly targets measurements in queueing networks (e.g., coping with unknown cross-traffic). Reactiveness of the network itself, namely, the dependence of latency on probing patterns (especially their history) has only received minor attention. Recently the Internet Engineering Task Force (IETF) adopted respective considerations in [127], which is however not finalized at present.

4.1.2 How to Benchmark Networks?

Benchmarking networks for a given application is straight forward. In such a case θ_p is given and the desired result is $\delta[n]{=}f(\theta_p, \psi, \phi)$. For measuring the respective quantity, the probing traffic has to be equal to the application traffic and any desired statistic (e.g., Empirical Cumulative Distribution Functions (ECDFs)) can directly be extracted from the plain measurement results. This topic is discussed in Chapter 5, where involved traffic models are specified for the accurate assessment of the network's delay response. Benchmarking reactive networks *without* any given application in mind is more complex. Obtained latency figures should ideally be valid for any kind of application (traffic pattern). In other words, the desired result is $\delta[n]{=}f(\theta, \psi, \phi)$, such that any $\theta{\in}\Theta$ could be plugged into the expression. Measuring the respective quantity bears two fundamental challenges:

- Θ is a high-dimensional vector space with one-sided infinite support; hence, performing an exhaustive measurement is infeasible.
- Ideally a measurement would yield a full delay model in the form of Eq. (2.5). In reality, however, such models would be too complex for an intuitive interpretation.

Both mentioned issues indicate the need for a reduced parameter set, which allows for generally valid conclusions while keeping the measurement effort bearable.
One way to define such a set would be to select a certain number of popular applications and to perform latency measurements with the respective traffic patterns. Each of the selected applications constitutes one point in the vector space Θ. This is illustrated in

Figure 4.1 for only two dimensions (i.e., PS $\pi[n]$ and IAT $\tau[n]$), where the green shapes indicate popular traffic patterns. The respective measurement result would consist of multiple delay figures (e.g., ECDFs), one for each traffic pattern, having the advantage of easy interpretation. The drawback of this approach is that the parameter space Θ is only sparsely sampled, with unevenly distributed measurement points (i.e., probing patterns). The immediate consequences are:

- Influences on latency are difficult to assign to a specific parameter of Θ. For example, for Voice over IP (VoIP) traffic it is impossible to distinguish between impacts caused by PS and IAT.
- The effects of even slight changes of parameters are hard to predict. For example, there is no guarantee that VoIP traffic with a specific rate will perform similar at a slightly different rate.

For the given reasons I refrain from pursuing the outlined approach and aim for a more general technique. Intuitively, a reactive network shows different latency performances for different traffic patterns, but if a broad range of traffic patterns is measured, there may be classes of patterns performing very similar. For the basic example outlined in Section 2.3.2.2 which is dominated by a threshold, such an effect can be observed for the two classes of traffic patterns strictly located at one side of the threshold. All the respective traffic patterns result in the same latency ECDF, namely, either Gaussian with mean latency of 9 ms or a shifted version with mean 11 ms. Further, the probing patterns crossing the threshold result in a mixture of both cases; hence, the resulting delay ECDFs are *bounded* on both sides by one of the Gaussian distributions. Summarizing, three classes of probing patterns are obtained (i.e., below, above and crossing the threshold), with fixed latency ECDFs assigned to the first two classes and a bounded region for ECDFs of the third class. This statement is stronger than any result obtained by single probing patterns, since it is generalizable to a whole class of probing patterns. The remaining challenge is to find a probing strategy to efficiently detect such classes. However, this probing strategy should cover multiple factors of Θ, not only IATs as in the example presented in Section 2.3.2.2.
Thresholds constitute clear boundaries between classes of specific latency behavior without causing variation within classes; hence, allow for an unambiguous assignment of probing patterns to a fixed number of classes. Real networks however might as well be influenced by effects causing continuous changes in latency. In such cases variations occurring within classes will cause a modeling error, which can be reduced by augmenting the number of classes. Nevertheless, recent work has shown that dominant thresholds exist in modern wireless networks [128] [1] [5], caused by various state-machines for single user sessions [22]. This fact justifies pursuing the outlined approach.

4.1.3 Technical Assumptions

The assumptions made below constitute the basis for a statistical sound measurement design [118]. They enable the determination of the timescale which yields the most significant measurement results. This is corresponding to a maximization of the impact of the traffic patterns Θ by simultaneously minimizing the influences of external effects Ψ and Φ. Since the assumptions do not necessarily hold for any arbitrary network, they have to be verified for each individual network under test. This is an integral part of the measurement procedure proposed in Section 4.1.4.
First, a temporal horizon T_Θ on the influence of the traffic pattern is assumed, such that all probing patterns θ with the identical recent history cause the same reaction of the

network. The horizon T_Θ can also be expressed in terms of number of previous packets N. Hence, only the last N packets cause variations in the delay:

Assumption 4.1

Let θ_i and θ_j denote probing patterns and T_Θ the temporal horizon of a reactive network. Further N is the smallest positive number such that $\sum_{m=n-N}^{n} \tau_k[m] \geq T_\Theta$, for both $k \in \{i, j\}$. If $\theta_i[m]=\theta_j[m]\ \forall\ n \geq m \geq n-N$, then

$$P\{\delta[n]|\theta_i, \psi, \phi\} = P\{\delta[n]|\theta_j, \psi, \phi\}.$$

This defines the minimum duration of a traffic pattern which is required to guarantee the fading of transient components (e.g., caused by former measurements). For all measurements presented below, T_Θ is in the order of few minutes.
The set Ψ contains all elements causing fast fluctuations of the latency. The fluctuations caused by the elements of Ψ are assumed to be wide sense stationary and, further, to have a fast decaying auto-correlation function. They are assumed independent for a temporal distance T_Ψ between samples which is bigger than a few seconds:

Assumption 4.2

Let ψ denote external influences causing fast fluctuations in delay $\delta[n]$. If N is a positive integer with $\sum_{m=n-N}^{n} \tau[m] \geq T_\Psi$, then

$$P\{\delta[N], \delta[n-N]|\theta, \psi, \phi\} = P\{\delta[n]|\theta, \psi, \phi\} \cdot P\{\delta[n-N]|\theta, \psi, \phi\}.$$

This is essential for the proposed measurement methodology, since it guarantees the accuracy of statistical evaluations in combination with short measurement periods (moderate number of multiples of T_Ψ, i.e., few tens of seconds).
The set of parameters Φ is causing constant or slowly changing delay. Thereby I assume the changes caused by Φ to be negligible within short time periods T_Φ (e.g., less than 15 min), which can be summarized in

Assumption 4.3

Let $\phi(t)$ denote external influences causing constant and slowly changing effects in the delay $\delta[n]$ at absolute time t. If T is a time interval with $|T| \leq T_\Phi$, then

$$P\{\delta[n]|\theta, \psi, \phi(t)\} = P\{\delta[n]|\theta, \psi, \phi(t+T)\}.$$

Again this assumption has important implications for the statistical evaluation of the measurements, since it allows for the explicit measurement of characteristics of $\delta[n]$ conditioned on a specific sample ϕ. Any comparative measurement performed right before or after the considered one (within less than T_Φ temporal distance) will experience delay conditions depending on the same parameter set. This is the basis for any significant comparison among different measurement setups.
Summarizing, the above assumptions indicate preferable measurement timescales for comparable and statistical significant results. For example, if two probing patterns shall be compared and the values T_Θ, T_Ψ and T_Φ are known for the network under investigation; then it is suggested to perform multiple measurement runs with both patterns in an interleaved fashion, such that each run is longer than T_Θ but the duration of the overall measurement is in the order ot T_Φ. For the following evaluation only packets with a temporal distance of more than T_Ψ should be considered.

Note again, that these assumptions have to be validated for each network or link (cf. Section 4.1.4.5).

4.1.4 A Generic Delay Assessment Strategy

An approach to achieve a universal understanding of the latency behavior of arbitrary reactive and non-reactive networks is described in the following. Coarsely speaking, this approach concerns four parameters of Θ, namely, (i) the actual packet size $\pi[n]$ and (ii) packet inter-arrival time $\tau[n]$, as well as (iii) the global data rate of the probing session $R{=}\frac{\overline{\pi}}{\overline{\tau}}$ and (iv) short term fluctuations in the instantaneous data rate $r[n]$. Those statistics are expected to reveal most of the influences on delay caused by Θ. Note, that they are inter-dependent (non-orthogonal), which is probably obscuring the root causes of certain effects. To clearly distinguish between such effects, multiple steps are required. They include iterative measurements and validations and are outlined in detail in the following.

4.1.4.1 Step 1: Defining a Region of Interest

In order to design a feasible latency probing scheme the parameters of Θ (i.e., PS $\pi[n]$ and IAT $\tau[n]$) have to be restricted, which is necessary since they have one-sided infinite support. Thereby a **region of interest** is defined, as depicted in Figure 4.1 by the red hexagon. Only values within this region are considered for further evaluation. Note, that the region of interest is not only valid for the present values $\pi[n]$ and $\tau[n]$, but for the whole session history $\pi[n{-}m]$ and $\tau[n{-}m]$, $\forall\ 0{\leq}m{\leq}n{-}1$. The boundaries of this region of interest are $\pi_{\min}$ and $\pi_{\max}$ for the PS, $\tau_{\min}$ and $\tau_{\max}$ for the IAT and $R_{\min}$ and $R_{\max}$ for the data rate. The respective numerical values are determined (restricted) by limitations of both the network under test and the measurement setup. Nevertheless, the boundaries should be selected such that the most prevalent applications are encompassed, as it is the case in Figure 4.1.

4.1.4.2 Step 2: Constant Bit Rate Probing

The next step is to perform a numerous amount of latency measurements, each for a different point in the vector space Θ (different probing patterns). The reason is to obtain a picture of the latency δ over the space Θ which is dense enough to yield minor discrepancies between adjacent probing patterns θ and θ', i.e., $\delta(\theta){\approx}\delta(\theta')\ \forall\theta{\approx}\theta'$. Note, that in the present context one measurement is *not* the transmission of a single probing packet, but the transmission of a whole probing pattern θ (i.e., point in Θ). Equivalently, a sample or measurement result does *not* refer to the latency experienced by one probing packet, but to a latency ECDF of $\delta(\theta)$ caused by a specific probing pattern θ.
To distinguish between effects caused by reactions on PS $\pi[n]$, IAT $\tau[n]$ and data rate R, without influence from fluctuations of the instantaneous rate $r[n]$, I perform multiple measurements with Constant Bit-Rate (CBR) probing patterns. CBR patterns are required since any kind of CBR traffic has the property of zero fluctuations in the instantaneous data rate $r[n]{=}R$; in other words, this specific dimension of Θ is always zero. The remaining three parameters are related according to $R{=}\frac{\pi[n]}{\tau[n]}$, thus, PS and IAT can freely be chosen but the mean data rate is defined by the tuple of PS and IAT. The aim is to maximize the number of patterns θ to be measured in this step, consequently the amount of time spent for one single pattern has to be minimized. Unfortunately, Assumption 4.1 requires that each session lasts longer than T_Θ, in order to guarantee the fading of transient components of θ. In contrast, Assumption 4.2 states

that independent samples can only be obtained for measurements which lie more than T_Ψ apart, to avoid possible correlations in the short term fluctuations ψ. This means that frequent measurements have to be taken but, simultaneously, yield poor quality. In order to circumvent this problem, the traffic patterns are injected in an interleaved fashion. This is only possible due to

Assumption 4.4

Let θ_c denote a CBR probing pattern with PS π_c, IAT τ_c and data rate $R_c{=}\frac{\pi_c}{\tau_c}$. Further, let θ denote a probing pattern with $\pi[n]$ and $\tau[n]$, and ϵ a constant close to zero. If $\pi[n]{=}10^{\pm\epsilon}\pi_c$, $\tau[n]{=}10^{\pm\epsilon}\tau_c$ and $\frac{\pi[n]}{\tau[n]}{=}R_c$ $\forall n{\in}\mathbb{N}$; then both patterns yield the same latency response

$$P\{\delta[n]|\theta,\psi,\phi\} = P\{\delta[n]|\theta_c,\psi,\phi\} \qquad \forall n{\in}\mathbb{N}.$$

By this assumption small changes in PS and IAT do not change the delay figure compared to CBR traffic, as long as the instantaneous data rate $r[n]$ is kept constant. Thus, a slow movement on diagonal lines (constant R) in the PS-IAT plane (see Figure 4.1) yields measurement results which are equivalent to CBR probing, as long as the change in PS or IAT is not bigger than ϵ within the temporal horizon T_θ (see Assumption 4.1). The big advantage is that by moving up and down such lines, each point is passed with a temporal distance of more than T_Ψ; hence, we obtain uncorrelated (high quality) delay values. Further, the obtained measurements are not only for single isolated points, but for a whole series of points arranged along a line with constant R; hence, the measurement resolution on this line is very high (one ECDF for each point on the line). It remains to optimize the movement along this line. Since it must be guaranteed that the neighborhood ϵ (e.g., 30%) of each point is not abandoned within T_θ, it is recommended to move with constant speed along this line. Since the neighborhood ϵ enters multiplicative in Assumption 4.4, the moving speed is constant in decades per time unit (logarithmic scale, cf. Figure 4.1). This implies that the same amount of time is spent within different intervals of IATs, e.g., the intervals 1 ms–10 ms and 1 s–10 s. Consequently, the amount of probing packets within an interval of short IATs is much higher than the amount of packets at long IATs. More precisely, the distribution of the probing packets follows a bounded Pareto distribution, cf. Appendix H, for both the PS and IAT. Thereby, the shape parameter $\alpha{=}1$, the lower bound L and the upper bound H are determined by the borders of the region of interest. Consequently, ECDFs obtained for short IATs have a much higher statistical significance than ECDFs obtained for long IATs.

The movement should inherit some kind of randomness, in order to avoid synchronization effects, cf. [129] [39]. There are various feasible approaches to introduce randomness, e.g., deploying a random walk or generating a Pareto distributed random process with strong correlations. In the present work random increments are used for this sake. The PS (or equivalently IAT) is raised for a small random increment until the end of the region of interest is reached. Afterwards, the direction is changed and the PS is decreased until the other end is achieved. Even for a short measurement duration any point on the line is reached according to the requirements (bounded Pareto density). This method is preferable, since the aim is to reduce the measurement duration, in order to stay within T_Φ (cf. Assumption 4.3) and obtain comparable results.

Finally, by measuring on multiple diagonal lines (multiple data rates R), a dense sampling of the PS-IAT plane can be achieved. The number of different rates is determined

by the trade-off between measurement duration and sampling resolution. For the measurements presented in Section D.3, see also Figure 4.2 (a), samples from three rates (lines) per decade are obtained, where the measurement duration for one line amounts to 10 min and the sampling of the whole region of interest to approximately 2 h. The increase in measurement resolution (in terms of number of sampled PS-IAT tuples) achieved by the presented method is estimated to a factor of 10–50 compared to primitive CBR probing. The reason is, to put it simply, that each of the roughly 50 points on a line can be measured in the course of one measurement run, instead of performing 50 separate runs.

4.1.4.3 Step 3: Clustering of Probing Patterns

The previous step results in a large number of latency ECDFs for CBR probing patterns, one for each measured PS-IAT tuple (e.g., some hundreds). In the present step this enormous amount of data has to be compressed in order to obtain only a handful of parameters which are easy to work with. This is achieved by arranging the ECDFs into groups with similar properties.

First, each ECDF is vectorized according to a logarithmic binning of latency. Logarithmic binning is feasible, since the involved latency ECDFs have positive support, and, further, advantageous since it (i) offers a high resolution at values close to zero and (ii) compresses the long tails exhibited by the delay (cf. [2]). For the evaluation presented in Section D.3 a binning from 100 μs to 10 s with η=100 bins/decade is used. Deploying delay ECDFs as characteristic parameters for the PS-IAT tuples implies normalization; hence, the loss of the information about the delay sample size (number of probing packets). This is required since the number is different for each tuple; the information on the statistical significance is however lost. For that reason I introduce a minimum sample size S required for a ECDF to be valid. Further, for comparing ECDFs with a different number of samples, the ECDF with more samples has to be sub-sampled, such that both ECDFs yield the same amount of samples.

After the vectorization step a clustering algorithm is applied to construct groups of PS-IAT tuples with similar ECDFs. The *K-means++* algorithm is thereby deployed [130]. This algorithm takes an arbitrary number of vectors to be clustered and a positive integer number K corresponding to the number of clusters. It calculates an optimum assignment of vectors to clusters, such that the overall Euclidean distance of all single vectors to the respective cluster mean is minimized. I determine the Euclidean distance $D_2(\mathbf{f}, \mathbf{g})$ in the specified vector space to correspond to a weighted version of Cramer's distance for random variables [131],

$$D_2(\mathbf{f}, \mathbf{g}) \equiv \sqrt{\frac{\eta}{\log_e(10)} \int_{0^+}^{\infty} \frac{1}{\delta} \Big(F(\delta) - G(\delta) \Big)^2 d\delta}, \tag{4.1}$$

where F and G denote the delay ECDFs, corresponding to the vectors $\mathbf{f}$ and $\mathbf{g}$, and η denotes the number of dimensions of the vectors per decade of δ. The output of the clustering algorithm are K cluster means (ECDFs in our case) and a respective mapping from each input vector (i.e., PS-IAT tuple) to a cluster. The number of clusters K is thereby an open input variable; however, the visual representation usually suffers from a high number of clusters. I suggest a maximum number of five clusters.

An example is shown in Figure 4.2, where the latency ECDFs of a WLAN network are clustered into four groups. The figures in the top row show the uplink, those in the bottom row the downlink direction. Observe, that in both directions only the packet size influences the latency; which is indicated by the horizontal cluster boundaries in

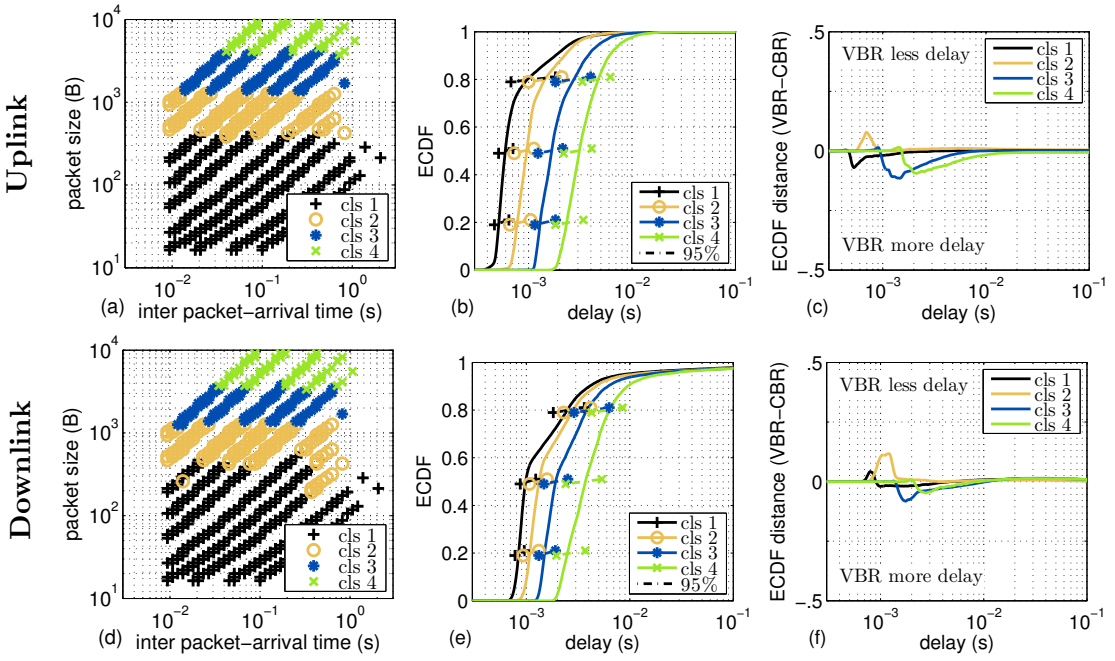


FIGURE 4.2: Latency behavior of WLAN in uplink (top, a–c) and downlink (bottom, d–f); $cls\,k$ denotes the kth cluster. Left (a, d): Clustering of latency ECDFs in the PS-IAT plane into K=4 groups. Center (b, e): Delay ECDFs for single clusters with 95% confidence intervals. Right (c, f): Distance between delay ECDFs of VBR and CBR traffic over delay, $F_{\mathrm{VBR}}(\delta) - F_{\mathrm{CBR}}(\delta)$. Both links show typical non-reactive behavior.

the rightmost figures. This is a distinguishing mark for non-reactive networks, where the only influences on latency are caused by the relation of packet size and maximum throughput. This assumption is supported by the almost equal separation (in linear scale) of the clusters in the PS; yielding equally spaced latency ECDFs which are horizontally shifted replicas with confidence intervals which are bordering at each other, i.e., Figure 4.2 (b).

4.1.4.4 *Step 4: Variable Bit Rate Probing*

This step generalizes the evaluation to variability in the traffic (i.e., changes in $r[n]$) which is achieved by measurements deploying VBR probing patterns. The aim is thereby to suppress the previously discovered effects (i.e., influences by present PS and IAT); hence, each cluster is treated individually in the following investigation. Separate traffic streams for each cluster are generated by randomly picking valid PS-IAT combinations for each packet. The respective probability is uniformly distributed over the whole cluster (in double logarithmic scale). Since each process consists of points from only one cluster, the respective latency measurements yield the same ECDFs as the cluster means, provided the network does *not* react on variability in the instantaneous data rate. If any obtained delay ECDF significantly differs from the respective cluster mean (especially if it lies outside the 95% confidence intervals), a relation between delay and fluctuations in data rate is established for the respective PS-IAT region. A different interpretation is that in such a case the recent history of the data stream influences the network's latency response.
The choice for sampling each packet randomly from the PS-IAT region is due to the resulting independence of consecutive packets; yielding a maximum variability and unpredictability in the instantaneous data rate. The traffic patterns in this step are in strong

contrast to the perfectly predictable patterns deployed in Step 2 (cf. Section 4.1.4.2). The ECDFs obtained in both steps constitute bounds for traffic patterns with intermediate behavior. Intermediate behavior can be interpreted as any positive correlation between consecutive packets, with the bounding cases of correlation close to 0 (i.e., randomly sampled from the cluster - unpredictable) and correlation close to 1 (i.e., CBR traffic - perfectly predictable). Note that this is only valid for traffic patterns which are limited to one single cluster area in the PS-IAT plane for the whole stream. If the boundaries to any other cluster would be crossed, there are inevitable influences from PS and IAT changing the latency behavior; hence, an inference on the influence of the session history is cumbersome.
Examples are given in Figure 4.2 (c, f). They show the distance of the ECDFs of the delays obtained from VBR probing (Step 4) and CBR probing (Step 2), $F_{\mathrm{VBR}}(\delta) - F_{\mathrm{CBR}}(\delta)$, over the delay δ. Thus, a straight line at 0 over the entire x-axis indicates that the latency ECDFs of VBR traffic and CBR traffic perfectly coincide. As observable, this is the case for WLAN both in uplink and downlink (i.e., the fluctuations are rather small). HSPA networks on the other hand, show a different behavior, see Figure D.4 (c, f). For example, for downlink Figure D.4 (f) shows a constant latency increase for the cluster Cluster 5 (+20%), corresponding to a constant right-shift of the ECDF for VBR compared to CBR, cf. Figure D.4 (e, f). The uplink, on the other hand, exhibits huge delay increases at higher quantiles of Cluster 2–3, corresponding to a longer and flatter tail of the ECDFs for VBR.

4.1.4.5 Step 5: Verification of Assumptions

The assumptions made in Section 4.1.3 have to be verified for each network under test, in order to guarantee correctness and significance of the obtained results. Therefore, I conducted several cross checks on the measurements.
A third type of probing pattern is required for this purpose, namely, Reference (REF) traffic. It consists of *primitive* CBR traffic, in the sense that PS and IAT are constant for each packet for a duration of several minutes (which is in contrast to the CBR variant used for measurements in Step 2). Thereby 16 different PS-IAT combinations are used, confer Figure 4.3, which meet the lines of constant rate spanned by the probing patterns deployed in Step 2.
Comparing latency responses for different patterns is aggravated by the fact that Φ causes long-term variations in the latency response. All measurements obtained at instances within an interval of less than T_Φ observe the same realization $\phi \in \Phi$, as desired for comparison. However, it is not possible to perform measurements for all required comparisons within T_Φ; hence, probes possibly experience different realizations from Φ. Equivalently, delay samples are correlated (in terms of Φ), depending on the temporal distance of the respective measurement. The most popular statistical tests however require independent samples, thus, cannot be used to compare the delay samples obtained by the different probing patterns directly.
Instead a different approach is pursued, namely, an evaluation on the basis of interleaved sessions. The sessions have a temporal distance of several hours, supporting independence due to Assumption 4.3. All samples within a probing session are combined to an ECDF and vectorized, distinguishing between the I vectors $\mathbf{s}_X[i]$ of sessions of type X and the J vectors $\mathbf{s}_Y[j]$ of probing sessions performed according to traffic type Y. Thereby, X and Y correspond either to VBR, CBR or REF traffic, respectively. The ground truth for the latency ECDF is defined as the average $\overline{\mathbf{s}_X}$. From these vector sets,

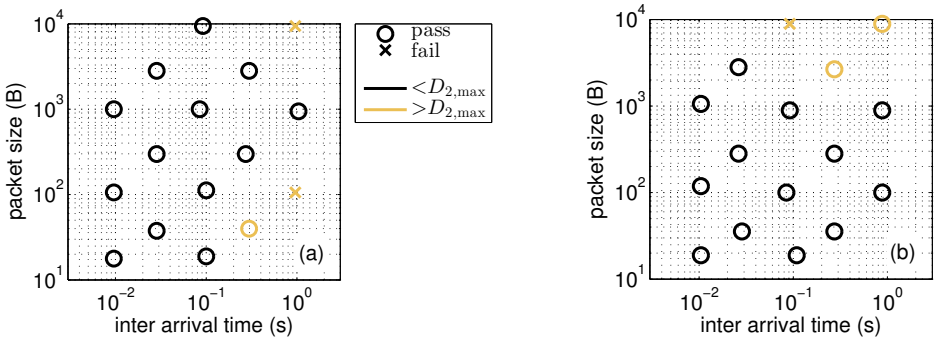


FIGURE 4.3: HSPA downlink, Provider 1: (a) Verification of Assumption 4.1 and Assumption 4.4 at 16 PS-IAT tuples. Both assumptions are accepted for tuples with black circles (positive CM-test, $D_2 < D_{2,\max}$). (b) Comparison of ECDFs captured in different weeks.

the set of distances to the ground truth $d_X[i]$ and $d_Y[j]$ are calculated, according to

$$d_X[i] = D_2(\mathbf{s}_X[i], \overline{\mathbf{s}_X}), \qquad d_Y[j] = D_2(\mathbf{s}_Y[j], \overline{\mathbf{s}_X}), \tag{4.2}$$

where $D_2(\cdot,\cdot)$ denotes the Euclidean distance, see Eq. (4.1). Both sets of distances can now be treated as independent samples; hence, statistical tests can be deployed to verify that both stem from the same distribution. I deploy the two-sample Cramer-von Mises test [132] in this case. Additionally, it is ensured that the distance between the average vectors of both sets $\overline{\mathbf{s}_Y}$ and $\overline{\mathbf{s}_X}$ is smaller than $D_{2,\max}$=0.25, which guarantees a high similarity of both ECDFs at the given number of dimensions per decade η=100.

Verification of Assumption 4.1 and Assumption 4.4: Section 4.1.4.2 mentions that Assumption 4.1 and Assumption 4.4 must both hold true such that the traffic pattern proposed in Step 2 (labeled CBR) yields the same latency figure as primitive CBR traffic (REF). Conversely, if both patterns yield the same latency figures both assumptions are asserted. Hence, both patterns are compared according to Eq. (4.2), by setting X to the REF pattern and Y to CBR. If the REF pattern yields the same latency ECDF as the CBR pattern in the immediate neighborhood of the respective point, both assumptions are proved.

In Figure 4.3 (a) the described method is illustrated. The figure shows a comparison between the REF probing pattern and the CBR probing pattern at 16 available PS-IAT combinations. The tested technology is thereby HSPA (Provider 1) in downlink direction. The circles and crosses indicate pass or fail of the CM-test, respectively; the corresponding color indicates the distance between the averages $D_2(\overline{\mathbf{s}_{\mathrm{CBR}}}, \overline{\mathbf{s}_{\mathrm{REF}}})$ of both traffic patterns, compared to $D_{2,\max}$. Hence, if the marker is a black circle, Assumption 4.1 and Assumption 4.4 are satisfied at this checkpoint. The assumption was tested for a line moving speed of 1 decade/min for all following measurements. Reducing the moving speed may be better for consistency, however, it would strongly increase the overall measurement duration. The mentioned speed yields ECDF congruence for most of the checkpoints for all networks, which implies correctness of Assumption 4.1 and Assumption 4.4.

Verification of Assumption 4.2: This assumption states that the fast fluctuations in the latency response caused by Ψ are independent from one another if they appear a certain period T_Ψ apart. It is verified by the aid of the REF traffic, for which the Autocorrelation Function (ACF) of the measured delay values experienced by consecutive packets is estimated. Due to the constant IATs (characteristic for the REF patterns),

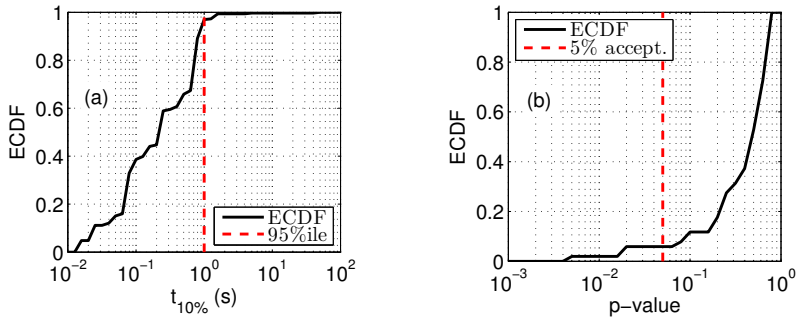


FIGURE 4.4: HSPA downlink, Provider 1: (a) Verification of Assumption 4.2, ECDF of $t_{10\%}$. (b) Verification of Assumption 4.3, ECDF of the p-value of the CM-test for sample sets with 5 min distance.

the ACFs can easily be translated from the packet index domain (n) to the temporal domain. The duration $t_{10\%}$ for multiple measurement sessions is determined, which is the duration after which the absolute value of the ACF is lower than 0.1. A respective evaluation is shown in Figure 4.4 (a) for HSPA, Provider 1, downlink. It is clearly visible that most of the ACFs decay to values below 10% after a very short period. Only 5% of all evaluated cases exhibit ACFs with $t_{10\%}$ bigger than 1 s. Assuming T_Ψ=10 s yields satisfactory results for all evaluated technologies and providers; hence, Assumption 4.2 is verified.

Verification of Assumption 4.3: The influences on the delay caused by slow effects Φ should be constant for any period smaller T_Φ. This assumption is verified by comparing samples from the CBR probing traffic of Step 2. Due to the incremental approach deployed for movements on the diagonal lines (cf. Section 4.1.4.2), nearly the same traffic pattern is injected several times by moving up and down on a single line. One line is probed for roughly 10 min, hence, by comparing the first 2 min with the last 2 min it is verified that $T_\Phi \geq 6$ min. Thereby I only consider the two highest data rates, R_c=100 kB/s and R_c=50 kB/s, since they yield a satisfactory high amount of samples in short periods (<2 min). Further, the data sets are sub-sampled such that the distance between single delay samples is bigger 1 s (i.e., $\geq T_\Psi$), what yields approximately 100 delay values per set. The resulting two sample sets are compared by deploying the two-sample CM-test. By doing so for multiple lines and multiple measurement runs, an ECDF of the p-values of the test is estimated. An example is shown in Figure 4.4 (b), where the HSPA network of Provider 1 is evaluated in downlink. The ECDF shows, that only few cases (i.e., <10%) exhibit p-values smaller than 5%, which is equivalent to not satisfying Assumption 4.3; thus Assumption 4.3 is regarded correct for the network under test.

Stability: The outlined verification steps are sufficient for the deployment of the present measurement methodology. Nevertheless, another verification is presented in Figure 4.3, concerning the *stability and reproducibility* of the presented methodology. Any delay assessment would be highly questionable if the results would change over time. As Figure 4.3 clearly demonstrates, this is not the case. The figure compares two measurement runs performed roughly one week apart. Almost all checkpoints yield coinciding ECDFs and only one checkpoint at the border of the region of interest fails the CM-test. Thus, strong evidence for the temporal stability of the approach is obtained.

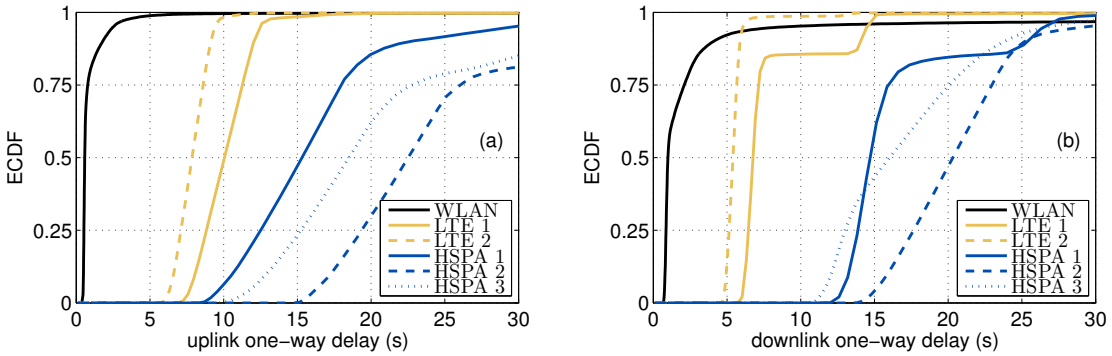


FIGURE 4.5: Comparison of OWD ECDFs of the *best* cluster (Cluster 1) among providers and technologies. (a) uplink, (b) downlink.

4.1.5 Fair Network Benchmarking

Benchmarking of networks requires the extraction of *one key value* for the whole latency response. This constitutes a clear ranking among multiple networks. The standard approaches (e.g., *ping*) consider thereby the average latency of one specific traffic pattern. I argue that such a value is insufficient to reflect the behavior of a reactive network and may yield unfair rankings (as outlined in Section D.3.3). Instead I propose to consult the **average latency of the best performing cluster as benchmark.** Alternatively, one could also evaluate the worst performing cluster. This representation is preferable over comparisons with a single traffic pattern, since the proposed ranking includes a multitude of traffic patterns.

Appendix D presents measurement studies for multiple public mobile cellular networks; revealing several interesting details on the network behavior. Those measurements, together with the study on WLAN presented above are now compared to each other on the basis of this **best-cluster benchmarking** methodology. A respective comparison is given in Table 4.1, which lists the median delay values for all assessed networks and link directions. The given clusters are ordered according to their average delay; hence, the networks are compared on the basis of the first column (i.e., Cluster 1). Notice, that the dimensions (in the PS-IAT plane) of the clusters change, cf. Figure 4.2, D.2–D.6 (a, d). A graphical representation of the best-cluster benchmark is provided in Figure 4.5. The figure depicts ECDFs of the best performing cluster (i.e., Cluster 1) for the various networks.

In general WLAN performs best with latency values below 1 ms, followed by Long Term Evolution (LTE) with One-Way Delays (OWDs) around 5–10 ms. As expected, HSPA networks perform worst with end-to-end delay values around 15-25 ms. LTE has met the design goal of halving the latency compared to HSPA [133] (cf. Table 4.1). Note however, that this comparison is biased for different (i) modems and (ii) network workloads. Comparing uplink and downlink direction shows that WLAN delays are lower in uplink, LTE exhibits less delay in downlink and HSPA shows similar values in both directions. The measurement campaign presented in this section was explicitly focused on the investigation of the influence of traffic patterns on the latency behavior. Consequently, the best-cluster benchmarks can be derived directly from the obtained measurements. This is of course a special case and cannot be assumed for any arbitrary problem statement; e.g., see Section 4.2. In such cases I recommend to perform a preliminary survey on the best traffic pattern (corresponding to the study presented in this section) and to choose one traffic pattern conforming to the best cluster for the principal survey. For example, the evaluation of the field trial presented in Appendix D shows, that CBR traffic with

a PS of less than 100 B and an IAT of less than 0.2 s is situated within the best cluster for all evaluated networks and technologies. Accordingly, such traffic is deployed for the study presented in Section 4.2.

4.1.6 Application Performance Estimation

From the measurement results presented in Section 4.1.4 and Appendix D, it is possible to estimate the latency performance of specific applications. This is a huge benefit compared to conventional latency measurements, which usually only allow for statements about the delay behavior of the probing traffic itself. In this section I provide a detailed explanation on how to derive application performance estimates by the aid of a fictive example. A graphical representation is provided in Figure 4.6.
Consider the measurement result given in the upper row of Figure 4.6 (framed in black). In Figure 4.6 (a) PS-IAT plane is depicted, being devided into two clusters with different latency ECDFs; see Figure 4.6 (b). Cluster 1 handles CBR and VBR traffic similarly, whereas for Cluster 2 VBR traffic experiences a higher delay; cf. Figure 4.6 (c). Several popular traffic patterns are included in Figure 4.6 (a) in green, for which the delay performance shall be estimated. Thereby, three cases have to be distinguished:

- **Compact traffic patterns**, such as *ping* and *VoIP*, are very similar to the CBR probing patterns deployed in Section 4.1.4 and can therefore be accurately evaluated. This is shown in Figure 4.6 (d). *VoIP* traffic is fully located in Cluster 1,

Technology	Provider	Direction	Traffic	Median OWD (ms)				
				cls. 1	cls. 2	cls. 3	cls. 4	cls. 5
WLAN		UL	CBR	0.57	0.85	1.26	2.46	
			VBR	0.59	0.83	1.38	2.74	
		DL	CBR	0.94	1.35	1.88	3.59	
			VBR	0.94	1.28	2.03	3.71	
LTE	1	UL	CBR	10.4	15.5	18.1	20	
			VBR	10.7	15.6	18.3	21	
		DL	CBR	7.06	7.37	8.28	9.11	
			VBR	7.03	7.31	8.38	9.23	
LTE	2	UL	CBR	8	11	15.1	16.5	
			VBR	8.1	11.1	15.1	16.9	
		DL	CBR	5.44	6.17	6.66		
			VBR	5.47	6.12	6.73		
HSPA	1	UL	CBR	16.0	21	30.2	58.4	111
			VBR	16.6	27	48	59.2	116
		DL	CBR	15.5	16.8	18.8	23.7	33.8
			VBR	15.6	16.7	18.7	26.2	41
HSPA	2	UL	CBR	25.8	32.1	45.1	72.9	
			VBR	27.6	41.9	62.7	96.8	
		DL	CBR	26.6	36.0	47.9		
			VBR	25.0	36.5	49.8		
HSPA	3	UL	CBR	19.4	25.2	34.1	51.3	
			VBR	33.3	30.3	36.1	44.7	
		DL	CBR	15.8	19.1	25.1	27.6	
			VBR	18.1	21.8	35.0	33.3	

TABLE 4.1: Latency (OWD) benchmarks for various mobile networks; *cls. k* denotes the *k*th cluster. Note, that the cluster dimensions change for different technology, provider and direction.

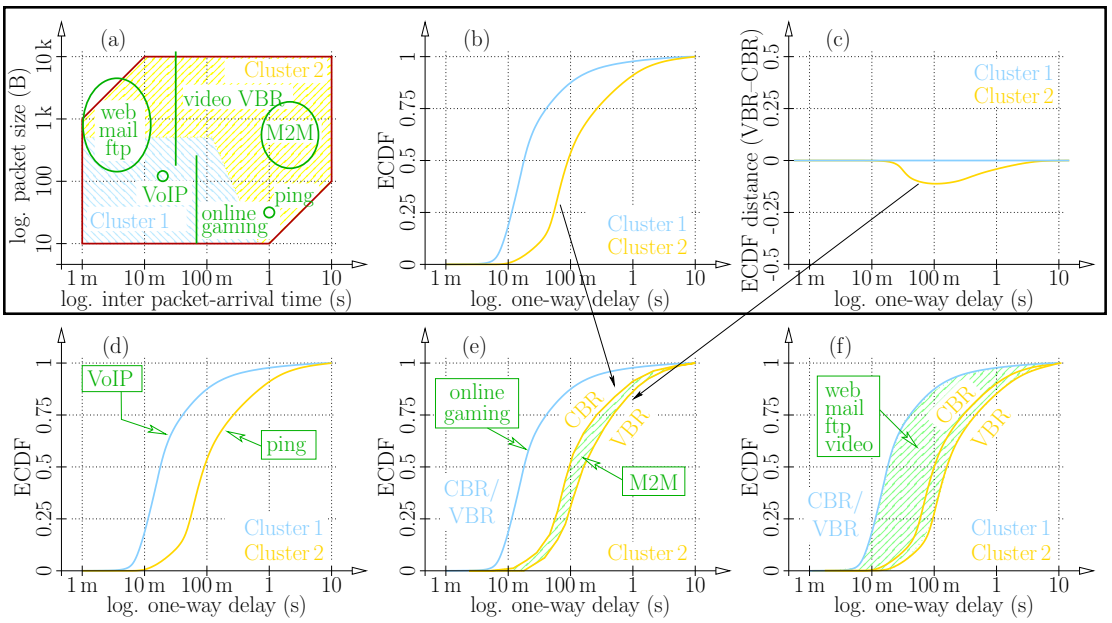


FIGURE 4.6: Application Performance Estimation: (top) Exemplary measurement results for an network with two clusters; additionally, different traffic patterns displayed in (a). Confer Figure 4.2 and Figure D.2–D.6. (bottom) Estimation of the delay performance of individual traffic patterns from the above figures.

hence, it is well approximated by the corresponding delay ECDF from Figure 4.6 (b). *ping* traffic is located in Cluster 2 and shows the respective latency ECDF.

- **Expanded traffic patterns within one cluster**, such as *online gaming* and *Machine-to-Machine Communication (M2M)*, can be bounded by the ECDFs for CBR and VBR traffic for the respective cluster. This is depicted in Figure 4.6 (e) where the shaded green area denotes the uncertainty for the respective delay figures. Thereby the delay ECDF corresponding to CBR traffic is obtained from Figure 4.6 (b), that for VBR from Figure 4.6 (c). For *online gaming* traffic both ECDFs perfectly coincide (straight line at zero in Figure 4.6 (c)); therefore the corresponding bounds on the delay performance are very tight and the uncertainty vanishes.

- The delay performance of **expanded traffic patterns over multiple clusters**, such as *video* and *web* applications, can be restricted to the area between the ECDFs with lowest and highest delay performance. Thereby all ECDFs have to be considered which belong to one of the touched clusters. In the present example the ECDF belonging to Cluster 1 shows the best delay performance, the ECDF of Cluster 2 VBR shows the worst. The performance of *web* and *video* traffic should conform to an intermediate case; however, this cannot be guaranteed, since traffic patterns with strong fluctuations which are crossing cluster borders have not been assessed in the course of the above measurement approach.

4.1.7 Summary and Criticism

In this section I presented a methodology to assess the delay behavior of reactive networks in a comprehensive fashion. Each network shows a very specific delay behavior,

just like a fingerprint; Appendix D emphasizes, that this is also true for network of the same technology but different providers. The presented methodology is based on a set of assumptions which must hold; if this is the case, the measurements can be sped up to a factor of 50. As a sanity check, I showed that the present methodology is very stable over time and yields reproducible results.
On the basis of the obtained delay-fingerprint of the network under investigation, the best-cluster benchmark can be determined. This benchmark is deduced from the lowest delay figure among all probing patterns. It is arguably more general than any benchmark obtained for an arbitrary but fixed traffic pattern and should therefore be used to compile rankings among networks. One of the corresponding best traffic patterns should further be used for any additional measurement study.
The high number of influences on delay (cf. Section 2.3.2, Table 2.3, [4]) complicates the measurement procedure; it will not be possible to inspect all parameters (dimensions), but some have to be fixed. The presented approach is tailored to determine influences of present and past PSs and IATs on the latency response of mobile networks. Consequently, not all parameters of Θ listed in Table 2.3 are considered within this section; neglected parameters are: (i) protocol related parameters and (ii) traffic on the reverse link. Further, unconsidered parameters of Φ are: (iii) physical location of the modem, (iv) version of the modem and (v) service level agreements. A respective implementation into the present measurement procedure would be straight forward, the execution of the resulting measurement procedure would, however, consume more time for the increased number of dimensions.
Another limitation is that the random measurements in Step 4 (cf. Section 4.1.4.4) only inspect *intra-cluster* variability. Hence, it is not possible to make any statements for inter-cluster behavior. The reason for not considering this type of effects is that (i) inter-cluster effects cause ambiguities in the assignment to respective causes, (ii) the number of combinations of clusters grows exponentially with the number of clusters. A respective investigation will be subject to future work.
Finally, our approach does not provide detailed models for the latency caused by traffic patterns, but bounds on the latency caused by specific traffic patterns. This is on the one hand preferable for the ease of interpretation and simplicity of the respective approach; the quality of the results, on the other hand, partly depends on the structure and number of clusters, as well as the closeness of the delay ECDFs obtained CBR and VBR probing.

4.2 Dissecting Influences of Network Components on Delay

Beside of the latency assessment for specific applications (as presented in Section 4.1.6 and Chapter 5), various other open questions concerning delay are worth to be considered. One of them is treated in the following, namely, the comparison of the performance of different network components and technologies. In the presented measurement study, an HSPA network is evaluated by comparing delay improvements due to two network upgrades. Both upgrades cover only parts of the mobile network and are not perfectly coinciding. Thereby I focus on two parameters which have been upgraded: (1) the uplink Transmission Time Interval (TTI) duration and (2) the Iub connection type. Two settings are possible for both parameters respectively (before and after the update), those are 2 ms and 10 ms for the uplink TTI duration and Asynchronous Transfer Mode (ATM) and Internet Protocol (IP) for the Iub connectivity.
The corresponding measurement setup is given in Section 3.4. Further, the deployed traffic pattern is corresponding to the *best-cluster* pattern as found in Section 4.1.5; in

this case the PS is ranging from 1–100 B and the IAT from 30–130 ms, uniformly distributed between these limits, respectively. In order to allow for a grading of the network updates, four different Base Stations (NodeBs), NB. 1–4, have been evaluated; all having different combinations of parameter settings. This corresponds to a full factorial experiment design [118].

4.2.1 Related Work

Latency assessments shown in popular literature are most often based on simulations (e.g., [134]) and estimations (e.g., [22]). This trend is observable for standardization documents, such as provided by 3rd Generation Partnership Project (3GPP) [135], for scientific publications [136] and for industrial reports [137] [138] [139].
Measurement-aided estimation techniques for OWD are also popular. One of the simplest techniques is based on the *ping* program, which measures the Round-Trip Time (RTT) from the client to a remote server in the Internet by sending *Internet Control Message Protocol (ICMP) echo requests* to the server. Some examples with focus on LTE and former 3GPP standards are found in [133] [140] [141] [142] [60] [59]. The OWD is thereby derived by assuming symmetric links and simply halving the RTT. However, the assumption of symmetric links does not hold true for mobile cellular systems, and our measurements confirm that the two directions display very different latencies.
True OWD measurements require the capturing and timestamping of data packets at both ends of the connection. This most often involves distributed and synchronized measurement nodes. The authors of [66] [65] [143] [144] present complete measurement tools for evaluation of communication links in terms of various metrics. Those are among others: jitter, packet loss, throughput and OWD. The timing is thereby provided by Global Positioning System (GPS) receivers and a custom software solution, the respective accuracy is estimated to below 100 μs. In [129] [128] measurements from multiple network operators are presented. The timestamping is done over a wired connection, with medium accuracy, however, retrospective clock synchronization enables satisfactory results. The authors use actively generated ICMP messages for latency assessment, in order to highlight the influence of the data generation method on RTT. In follow-up work, the authors focus on the impact of (i) time slotted transmission [145] and (ii) specific traffic patterns [146] on delay measurements. The authors of [62] [63] assessed the performance of different HSPA networks in terms of OWD. Their measurement devices are similar to those deployed within this work, whereas the timestamping accuracy is estimated to below 100 ns [86, pp. 97–98]. The difference to the present work is the generated traffic, i.e., the authors use the *ping* program.
Latency analyses with direct access to mobile network components are rare in literature. Some examples are [56] [57] [147], which perform passive large-scale RTT measurements by monitoring the TCP-handshakes of mobile users. Further, [67] [148] [149] report OWD measurements within the core network of Universal Mobile Telecommunication System (UMTS). Thereby the authors focus on anomaly detection. The passive monitoring system of those measurement campaigns has been partly reused for the present work.

4.2.2 Measurement Results

In the following, delay contributions of individual network components are analyzed. The involved network components are: (i) User Equipment (UE), (ii) NodeB, (iii) Radio Network Controller (RNC) and (iv) Gateway GPRS Support Node (GGSN). The uppermost branch of Figure 3.2 shows the deployed measurement probes (i.e., Probe 1–5),

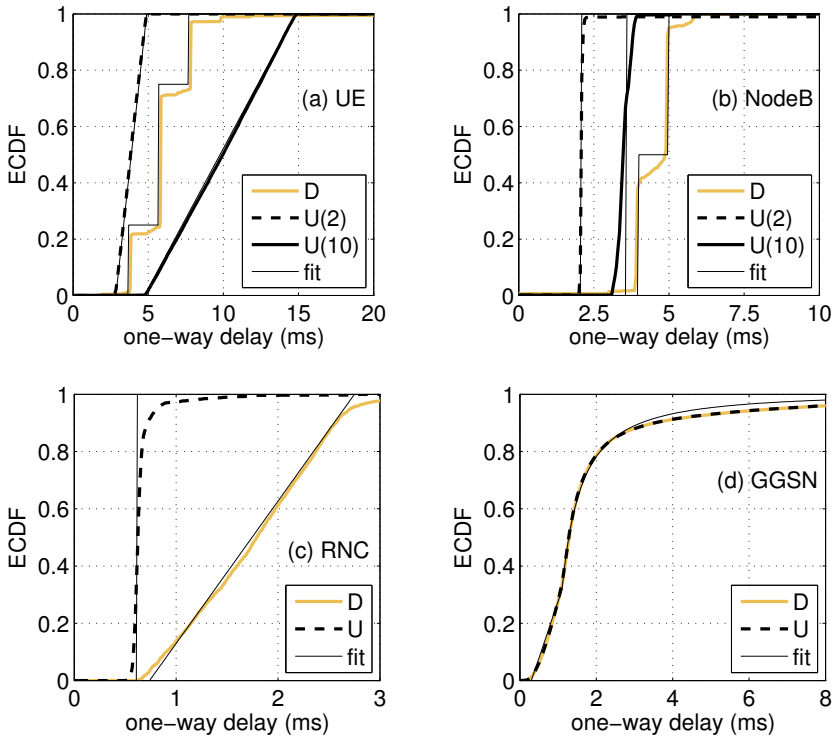


FIGURE 4.7: ECDFs of latency introduced by single network components for best-cluster traffic. Only IP backbone interfaces are considered, *D* denotes downlink, *U(2)* uplink with 2 ms TTI, *U(10)* uplink with 10 ms TTI and *fit* the respective fitted model (cf. Table 4.2). See Figure 3.2 for an illustration of the network.

where the contributions of single components are assessed by considering timestamps from both adjacent probes (e.g., the delay contribution of the RNC is deduced from Probe 3 and Probe 4). I refrained from taking the contribution of the Internet connection into account (e.g., results from Probe 6 are discarded), the reason being that it is not part of the *mobile cellular network*. ECDFs of the four latency contributions are illustrated in Figure 4.7 (the order of the plots corresponds to that of the network components). Packets which experienced retransmissions have been removed (and modeled separately, cf. Section 4.2.3); the reason being twofold: (i) the exact latency caused by retransmissions δ_{Retrans} is standardized [22] and (ii) retransmission ratios may drastically change with the radio conditions; hence, the ratios encountered within the measurements are not representative.

Figure 4.7 (a) shows the ECDFs of the latency caused by the UE, named δ_{Modem}. Thereby, the TTI-duration δ_{TTI} is excluded, for both, the uplink and the downlink case; consequently for uplink $\delta_{\text{Modem,U}} = t_{\text{P2}} - t_{\text{P1}} - \delta_{\text{TTI}}$, whereas for downlink $\delta_{\text{Modem,D}} = t_{\text{P1}} - t_{\text{P2}}$. The timestamps obtained at Probe 1 and Probe 2 are named t_{P1} and t_{P2} respectively. The figure illustrates, that a substantial amount of latency is generated in the UE, which is reasonable because of the limited energy and processing power.

The latency caused by the NodeB δ_{NodeB} is presented in Figure 4.7 (b). Again the TTI duration is excluded, as well as the delay caused by the ATM connection. This results in $\delta_{\text{NodeB,U}} = t_{\text{P3}} - t_{\text{P2}} - \delta_{\text{ATM}}$ for uplink and $\delta_{\text{NodeB,D}} = t_{\text{P2}} - t_{\text{P3}} - \delta_{\text{TTI}} - \delta_{\text{ATM}}$ for downlink. Note, that the latency introduced by the NodeB is relatively small compared to respective estimates found in literature (e.g., 5–7 ms [22]). Especially in the uplink direction, packets are forwarded unexpectedly fast, even though the detection and demodulation has to be performed.

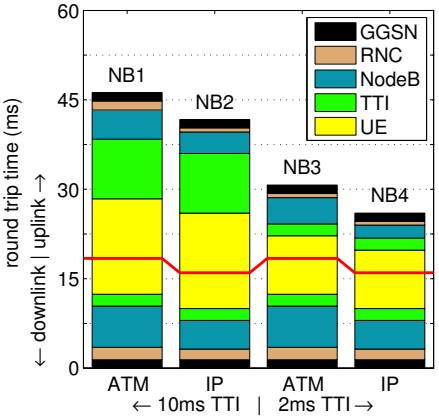


FIGURE 4.8: Median RTT for packets with PS smaller 100 B and IAT of 30–130 ms. Four cells are considered each of which reveals a different setup; i.e., ATM and IP cell connectivity and 10 ms and 2 ms TTI.

The ECDF of the delay introduced by the RNC ($\delta_{\text{RNC}}=|t_{\text{P4}}-t_{\text{P3}}|$) is show in Figure 4.7 (c). In the uplink the delay is constant at 0.6 ms, whereas in the downlink it is uniformly distributed in the range of 0.6 to 2.6 ms; the reason being that the core network is not synchronized to the UMTS Terrestrial Radio Access Network (UTRAN); hence, packets need to be aligned to the TTI slots in the RNC. For complete statistics, see [1].
Figure 4.7 (d) shows the empirical ECDFs of the delay introduced by the GGSN, $\delta_{\text{GGSN}} = |t_{\text{P5}}-t_{\text{P4}}|$. Both ECDFs of uplink and downlink latency almost perfectly match. They exhibit long tails (as expected for the highly aggregated link) and are, surprisingly, independent of the packet size. Their unconventional shape leads to the assumption that δ_{GGSN} is influenced by involved internal control mechanisms. Note, that *every* ECDF in Figure 4.7 reveals that the downlink is slower (or equal) than the uplink (for the 2 ms TTI case).
A breakdown of the overall RTT into contributions of single components is presented in Figure 4.8. The exact numerical values leading to this figure are tabulated in [2]. Thereby, median values have been considered, in order to reduce influences from the tails of the delay distributions (see Figure 4.7 (d)) and the influence of multiple modes of the distributions. Note however, that the sum of the single median values does not perfectly match the median of the sum of the RTT as Figure 4.8 may suggest (cf. Table 4.3). This is due to remaining correlations between the delay contributions of the single components. Figure 4.8 exactly dissects the latency shares of the network components of the HSPA network; thus, can be used to evaluate the respective estimations available in literature. Examples are the estimations found in [22]. Furthermore, Figure 4.8 highlights the latency share between uplink and downlink.
Observe, that both network upgrades have a decent impact on the delay performance. Thereby, the reduction in TTI duration yields higher performance improvements (i.e., ∼14.5 ms) than the transition from ATM to IP connectivity on the Iub interface (i.e., ∼5 ms). The overall delay reduction amounts to 20 ms (i.e., from 46 ms to 26 ms), corresponding to a performance improvement of roughly 43%.

4.2.3 Modeling

I provide a model for RTT based on several models for delay contributions of individual network components. The suggested models are kept simple in favor of reproducibility. Although there is a slight mismatch between models and measurements (cf. Figure 4.9),

I consider the modeling accuracy satisfactory. In some cases delays are assumed constant, although they exhibit a certain variance (e.g., $\delta_{\mathrm{RNC,U}}$). Those spreads are minor compared to the overall RTT; hence, not encompassed by the model.
The overall model is based on the assumption that the complete RTT, δ_{RTT}, is simply obtained by the sum of the latencies produced by the individual network components, both in uplink and downlink.

$$\begin{aligned}\delta_{\mathrm{RTT}} =& \delta_{\mathrm{U}} + \delta_{\mathrm{D}} \\ \delta_{\mathrm{U}} =& \delta_{\mathrm{Modem,U}} + \delta_{\mathrm{TTI,U}} + \delta_{\mathrm{Retrans,U}} + \delta_{\mathrm{NodeB,U}} \\ &+ \delta_{\mathrm{ATM}} + \delta_{\mathrm{RNC,U}} + \delta_{\mathrm{GGSN}} + \delta_{\mathrm{Size,U}} \\ \delta_{\mathrm{D}} =& \delta_{\mathrm{Modem,D}} + \delta_{\mathrm{TTI,D}} + \delta_{\mathrm{Retrans,D}} + \delta_{\mathrm{NodeB,D}} \\ &+ \delta_{\mathrm{ATM}} + \delta_{\mathrm{RNC,D}} + \delta_{\mathrm{GGSN}} + \delta_{\mathrm{Size,D}}.\end{aligned} \tag{4.3}$$

Here δ_U and δ_D denote the uplink and downlink OWD, respectively. Concerning single packets, Eq. (4.3) holds clearly true (cf. *additivity property*, Definition 2.1). However, by interpreting the delays δ_i as inter-dependent random variables (or processes), the equations are not valid any longer. The reason is the correlation between single latencies; e.g., between uplink δ_{U} and downlink δ_{D} due to synchronization effects [129]. Nevertheless, I model the delay components as independent, in order to reduce complexity. This degrades the above equation to an approximation, whose quality will be assessed in Table 4.3. In order to mitigate the strongest correlations, artificial latency factors are introduced in Eq. (4.3), namely:

- δ_{TTI} for the TTI duration.
- $\delta_{\mathrm{Retrans}}$ for packet retransmissions at the air-interface.
- δ_{ATM} for an Iub-interface based on ATM links.
- δ_{Size} for the packet size.

Statistical models for every latency factor are given in Table 4.2. All values in the table are given in milliseconds. A visual comparison of these models to their measured

	Component-wise delay models (ms)		
	Uplink		Downlink
	2 ms TTI	10 ms TTI	
δ_{Modem}	$\mathcal{U}(2.9, 4.9)$	$\mathcal{U}(4.8, 14.8)$	$3.7 + 2{\cdot}\mathcal{B}(2, \frac{1}{2})$
δ_{TTI}	2	10	2
$\delta_{\mathrm{Retrans}}$	16 $\mathcal{G}eom(1{-}P_{\mathrm{R,U}})$	40 $\mathcal{G}eom(1{-}P_{\mathrm{R,U}})$	12 $\mathcal{G}eom(1{-}P_{\mathrm{R,D}})$
δ_{NodeB}	2.1	3.6	$4 + 1{\cdot}\mathcal{B}(1, \frac{1}{2})$
δ_{ATM}	2.3 (ATM) or 0 (IP)		
δ_{RNC}	0.6		$\mathcal{U}(0.8, 2.8)$
δ_{GGSN}	$BU + (1{-}B)P$ with $B \sim \mathcal{B}(1, \frac{1}{3}),\ U \sim \mathcal{U}(0.4, 1.2),\ P \sim \mathcal{GP}(0.75, 0.55, 1.2)$		
δ_{Size}	$\pi[n]\,/\,C_{\max}$		

TABLE 4.2: Statistical models for delay contributions of single network elements and other main influences. A evaluation is given in Table 4.3 and Figure 4.9, including the values $P_{\mathrm{R,U}}$ and $P_{\mathrm{R,D}}$. For definitions of distributions see Appendix H.

counterparts are given in Figure 4.7. The thin dark lines therein, labeled *fit*, correspond to the respective models. An explanation of the models follows below.

- **Modem:** The delay δ_{Modem} introduced by the modem is determined by evaluation of Figure 4.7 (a). Observe that the uplink delays for both 2 ms and 10 ms TTI are uniformly distributed. This is due to the required synchronization to the TTI slots. The downlink delay consists of discrete values, spaced 2 ms apart, which indicates that both input and output interfaces (application interface and air interface) are synchronized. The cause is the synchronization of the USB polling interval (1 ms) to the TTI (2 ms), which is an integer multiple of the former; hence, the time between USB polling and beginning of a TTI is always constant. Since the USB polling interval within our setup was 1 ms, one would anticipate discrete values with 1 ms spacing. However, the spacing is 2 ms, what is explicable only by an internal pipeline structure of the modem, working at the TTI basis, which prevents any data to be available at the USB port before the next TTI is completed. This behavior is mapped onto a binomial distribution $\mathcal{B}(2, \frac{1}{2})$ with two trials and probability one half, multiplied by an interval of 2 ms and an offset of 3.7 ms.

- **TTI:** The TTI introduces an extra delay of 2 ms or 10 ms in the uplink and 2 ms in the downlink, these values are constants. They are the time the packets require to be transmitted on the air interface.

- **Retransmissions:** The retransmissions at the air interface introduce discrete values of delay, depending on the TTI, the direction of transmission, and the respective probability of retransmission (i.e., $P_{R,U}$ for the uplink and $P_{R,D}$ for the downlink). The resulting models consist of a time interval (tabulated in the corresponding standardization document [150]), multiplied by a geometric distribution $\mathcal{G}eom(1-P_{\text{R}})$, which is influenced by the retransmission probabilities. The geometric distribution indicates the number of successive fails of a Bernoulli trial until the first success; hence, accounts for multiple retransmissions.

- **NodeB:** The delay introduced by the base station can be assessed by studying the ECDFs in Figure 4.7 (b). In the uplink the delay ECDFs shows steep raises. Therefore, the delay is interpreted as constant, both for the 2 ms and 10 ms TTI case. The situation is different for the downlink. The ECDF is composed by two discrete values, both with roughly the same intensity. This suggests that the Iub interface is synchronized to the TTI frames but works on a 1 ms time base. Consequently, the NodeB has to buffer packets which arrive at odd frames, in order to ensure alignment to the TTI. I model this behavior by a constant delay of 3.6 ms in combination with an interval of 1 ms multiplied by a Bernoulli trial of probability one half $\mathcal{B}(1, \frac{1}{2})$.

- **ATM:** If the ATM technology is deployed for the Iub link, an additional constant delay of 2.3 ms is experienced both in uplink and downlink direction.

- **RNC:** This network component causes a constant delay in uplink of about 0.6 ms. Figure 4.7 (c) suggests an additional variability around this value which, however, the model does not account for. The variation is only noticeable due to the high temporal resolution of this figure, leading to an overvaluation of this effect. The respective standard deviation is around 0.05 ms, which is considered negligible compared to the contribution of other components. In the downlink direction an uniform distribution due to synchronization is encountered; more precisely, the

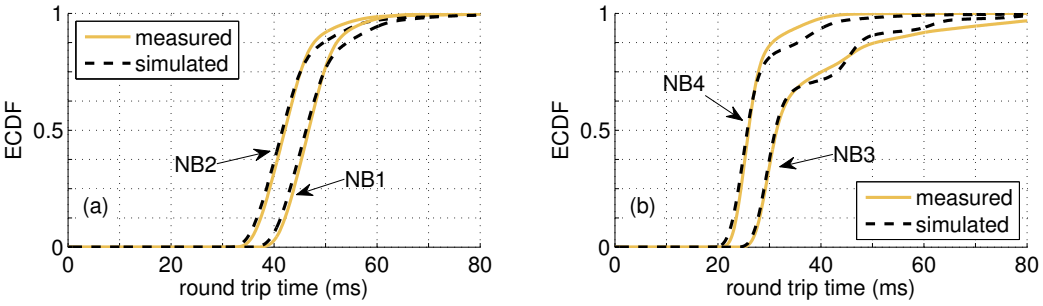


FIGURE 4.9: Evaluation of the models proposed in Table 4.2 by comparison to measured statistics. (a): The 10 ms uplink TTI case (NB2: IP, NB1: ATM). (b): Evaluation of the 2 ms uplink TTI case (NB4: IP, NB3: ATM). Notice the performance decreases at high retransmission rates (ATM, 2 ms TTI). Numerical values are given in Table 4.3.

NodeB	Conn.	$\delta_{\mathrm{TTI,U}}$	$P_{\mathrm{R,U}}$ (%)	$P_{\mathrm{R,D}}$ (%)	$D_{\max}$ (%)
NB1	ATM	10 ms	0	10	5.7
NB2	ATM	2 ms	24	3	3.8
NB3	IP	10 ms	0	10	4.9
NB4	IP	2 ms	4	7	7.0

TABLE 4.3: Evaluation of the models proposed in Table 4.2 by comparison to measured statistics. $D_{\max}$ denotes the maximum distance of simulated and measured ECDFs. Corresponding graphs are shown in Figure 4.9.

packets coming from the core network have to be aligned to the TTI slots, which is the heartbeat of the UTRAN.

- **GGSN:** The GGSN shows an unexpected behavior, see Figure 4.7 (d), namely, the statistics for uplink and downlink delay are perfectly matching. Furthermore, both distributions show a strong tail. The third remarkable property is that the ECDFs seem to consist of two regions with clearly distinguishable behavior, namely, below and above 1.2 ms. These properties can be explained by involved internal mechanisms. For modeling the delay of this component a bipartite distribution is deployed, consisting of a uniform distribution $\mathcal{U}(0.4, 1.2)$ for the body of the distribution (with probability of $\frac{1}{3}$) and a generalized Pareto distribution (cf. Appendix H) for their tail (with probability of $\frac{2}{3}$).

- **Size:** Finally, the delay evoked by the packet size and the maximum throughput $C_{\max}$ at the network interfaces is considered. Each interface in the whole data route of the network contributes its part to this value. This can be simplified by defining the throughput limitation on one interface as dominant (lowest possible throughput, e.g., air interface). Hence, the delay can be modeled as the ratio of the packet size and the dominant throughput, what, however, slightly underestimates the resulting latency.

These models are evaluated by comparing simulated RTT statistics to the measured RTT distributions, cf. Figure 4.7. Thereby the probabilities for uplink and downlink retransmissions have been determined by numerical optimization. An evaluation is given in Table 4.3 in terms of maximum absolute ECDF distance $D_{\max}$ between measured and simulated curves. A visual assessment of the model quality is provided in Figure 4.9. Note that the strongest discrepancies between measurements and simulations arise at

the bending at high values of the ECDFs. This inaccuracy is presumed to be caused by the latency of the GGSN and its respective model. Recall, that the model presented in Eq. (4.3) is only exact if all delay contributions from the network components are independent. The measurements, on the other hand, suggest a certain correlation; nevertheless, confirm that the assumption of independence is a reasonable approximation, given the additional delay values which have been introduced.

4.2.4 Summary and Criticism

In this section I presented delay measurements from a live public HSPA network. The focus was thereby on dissecting the delay contributions of each network element: i.e., UE, NodeB, RNC and GGSN. A detailed breakdown of the respective contributions is given, both in uplink and downlink. Further, the effect of network upgrades on delay has been studied. Two types of upgrades have been considered: (1) the upgrade from 10 ms uplink TTI to 2 ms and (2) the upgrade from ATM Iub connections to IP. Both upgrades yield a significant delay reduction, which amounts in total to about 43%.
In addition I built empirical models for the individual network components. The focus was thereby on the respective simplicity and reproducibility; nevertheless, an evaluation of the quality of the models shows good accordance with reality. The assumption that all delay components are independent random variables, made in Eq. (4.3), did only hold approximately (cf. Figure 4.9). The respective model could be improved by allowing for correlations between the variables.
The experimental design considers only a number of four NodeBs. Consequently, the measurement uncertainty caused by secondary effects cannot be suppressed by averaging. Similarly, the measurement duration was roughly 1–2 h per cell, which is not sufficient to average over daily variations.
The measurement setup for the air interface has two shortcomings (cf. Section 3.4): (i) the packets in the uplink direction are not decoded, but the timing is derived from the variations in transmit power of the UE [1]. The assignment of the beginning and ending of packets may thereby suffer from certain ambiguities due to retransmissions and transmit power control. (ii) the packets in the downlink direction were not captured during the measurements, but assessed offline by separate measurements with special equipment. The respective statistics are merged in a post-processing step [2]. Both shortcomings have been accepted in favor of the reduced costs of the measurement setup, both in terms of time and money.

Chapter 5

Traffic Models

Latency figures of mobile cellular networks are very sensitive on the injected traffic patterns, as previous chapters have demonstrated. If the delay performance of a specific application shall be assessed for such a network, it is therefore questionable to inject arbitrary traffic patterns during the measurement. Instead, I recommend to inject either real traffic or traffic patterns which accurately resemble it.
In this chapter several traffic models and modeling approaches are presented. They allow for the synthetic generation of network traffic patterns with strong resemblance to their real counterpart. Furthermore, the models enable the generation of patterns of arbitrary length with low complexity. They are especially suited for simulation platforms; respective implementations for the Open Air Interface platform [151] are available.
Section 5.1 describes a method for the accurate modeling of generic *network source traffic*. This kind of traffic refers to the data patterns encountered at the end-host. The proposed method shows very good results for real-time traffic, such as video and online-gaming. In Section 5.2 a modeling approach for *Machine-to-Machine Communication (M2M) traffic* is outlined. This kind of traffic is characterized by the presence of a large number of M2M devices which behave in a synchronized fashion. Standard traffic models (such as the one presented in Section 5.1) have problems with the simultaneous generation of traffic patterns for enormous amounts of devices. The presented model, on the other hand, is capable of dealing with this issue. Finally, Section 5.3 presents a traffic model for *cellular background traffic*. This model provides realistic load scenarios for cellular networks. Those can be used for simulations or measurements, in order to assess the impact of secondary users on the performance of the primary user in a controlled way.

5.1 Transformed Gaussian Models

It is widely agreed that data traffic can be modeled by stochastic process(es) [30] [152]. However, the physical quantity of the processes varies with the field of application. Classic queuing theory, for example, deals with arrivals of customers [24]. Further, from a packet networking perspective, the short-term data rate may be of interest [153], since it allows for the determination of buffer sizes. In video modeling the size of video-frames is commonly used to characterize traffic [154] [155], because frames are issued with constant rate, thus, the frame size allows for a complete characterization of the data stream. Finally, by modeling online-gaming traffic both, the Internet Protocol (IP) Packet Size (PS) as well as the Inter Packet-Arrival Time (IAT), are jointly considered as quantities for traffic modeling [156]. Throughout this section I will stick to this last approach, because it is generally applicable to Internet source traffic. Nevertheless, the

methods presented in this section are generic and can similarly be applied to random processes of any other physical quantity.
By characterizing recorded network traffic and providing models for reproduction and emulation, it is of interest to work with models which: (i) inherit as much statistical information about the original processes as possible, (ii) are flexible enough to be applied for a variety of different traffic types (e.g., video, web, background) and (iii) are parsimonious in terms of model parameters. In this section I address these needs by proposing a variant of Transformed Auto-Regressive Moving-Average (TARMA) processes, which is able to jointly characterize a broad range of Cumulative Distribution Functions (CDFs), Auto-correlation Functions (ACFs) and Cross-correlation Functions (XCFs) for multiple random processes as they typically appear in this context. Hence, three different statistical measures can be captured which are known to impact the network performance. An evaluation of the framework proves it to be flexible and parsimonious.
It has been shown by several authors that neither the CDF nor the ACF of a given random process are able to characterize all aspects which influence the respective queuing behavior [157] [158] [159] [160] [161]. In order to reinforce the validity of this statement, I demonstrate it by a simple example, where the discussed features are added step by step to simulated data traffic. A queue with a single server and constant service time per byte is simulated (e.g., a communication link would show such a behavior). The input process consists of a packet stream, of which the statistical properties of the PS and the IAT are varied for different simulation runs by using the method presented below. The Complementary Cumulative Distribution Functions (CCDFs) of the queue length are depicted in Figure 5.1. The different curves show the following scenarios: (i) constant IAT and gamma distributed PS, e.g., encountered for video traffic [162] [163], causing the shortest buffer-length, (ii) additionally, the constant packet IAT is changed to an exponentially distributed IAT, (iii) auto-correlations are introduced to the PS process by an AR(1) filter, (iv) furthermore, the same auto-correlation structure is introduced to the IAT and (v) finally, strong negative cross-correlation between PS and IAT is imposed to the processes, which is causing the longest buffers, (i.e., big PSs coincide with small IATs). It is clearly visible in Figure 5.1 that CDFs as well as ACFs and XCF have an impact on the queue length, for which variations over more than two decades are observed. Conversely, for example, if the rightmost curve (v) would correspond to original measured traffic and one would model it by fitting only its CDFs (i.e., neglecting ACFs and XCF, corresponding to a renewal process), the queueing response of the model would correspond to the second curve from the left (ii); hence, the actual queueing response would be underestimated by about two magnitudes. Similar examples can further be constructed for modeling only the ACFs and only the XCF.
Another example, underlining the importance of modeling CDFs, ACFs and XCFs jointly is given in Section 2.3.2.3. It clearly shows that each of the mentioned statistics influences the experienced latency in an High Speed Packet Access (HSPA) network on its own. This justifies the claim for a traffic model which catches all three of the above described statistical measures for multiple physical quantities.

5.1.1 Related Work

Traffic modeling is a topic of active research since roughly three decades. Summaries are given in [30] [152]. Most modeling approaches have in common that they model traffic streams as one or more stochastic processes. One simple approach is to assume one renewal process to be sufficient for an accurate representation of all relevant properties

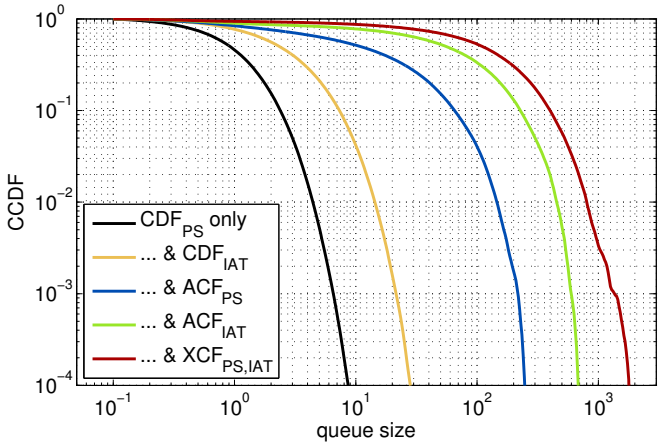


FIGURE 5.1: Queueing response on data traffic for different statistical properties of the respective packet sizes and packet inter-arrival times, U=0.8.

of the traffic. In this case only the distribution of the process can be modeled. The respective methods for parameter estimation and synthetic traffic generation are well established in literature [164] [165] [166] and implemented in simulation tools. The distributions exhibit characteristics which impact on the network and queueing response of the random processes; hence, respective modeling is justified. Such characteristics may be, for example, heavy-tails [167] [168] [169] or, more general, skewness [162] [155] [170] [163].

The assumption of independent identically distributed (i.i.d.) random processes, however, is often violated for network traffic [171] [172] [173] [174]. This is especially the case if long-range dependencies and self-similarities occur [175] [168] [176]. Besides non-stationary modeling approaches (which are beyond the scope of this work), there are two standard classes of models for such temporal dependencies; namely, regression models (i.e., Auto-Regressive Moving-Average (ARMA)) and Markovian models (i.e., Markov Modulated Poisson Process (MMPP)). ARMA models rely on linear filter theory and benefit from a long history with comprehensive literature [166] [177]. However, they are not able to handle broad ranges of distributions, but only a limited class (e.g., the normal distribution). Markov models, on the other hand, are a very general tool, leading in its basic form to renewal theory, cf. Section 2.3.1.2. They can be used to model various discrete distributions and auto-correlations; whereas for obtaining continuous distributions, as mostly required for traffic modeling, Markov models have to be extended to hidden Markov models or Markov modulated processes (Markovian models).

A further property of interest for traffic modeling is to capture cross-correlations in network traffic. This idea is natural when migrating from one to multiple random processes (i.e., physical quantities). Examples would be (i) packet networking, where size (PS) and arrival time (IAT) have to be considered jointly, (ii) video streaming, where different video frame types (e.g., I,P and B in MPEG4) can be thought of individual random processes, and (iii) multiplayer online gaming, where one server issues multiple correlated packet streams to the individual players. Literature provides few examples for traffic models where cross-correlations were considered.

Summarizing, standard models are not able to represent a broad range of CDFs, ACFs and XCFs jointly. This results in a variety of data traffic models, each of which designed for either a specific application or a specific network type. On the other hand, there are only few modeling approaches which are capable of jointly representing the above mentioned statistics. Those can be summarized in three categories: (i) Markovian models,

	Markovian	TES	TARMA
Statistical measure	CDF, ACF, XCF		
Heavy tailed distribution	approximations		
Long correlation	approximations		
Analytically tractable	✓	✓	✓
Queueing theoretic results	✓		
Separable fitting problems		✓	✓
Parsimonious	✓		✓
Automatic fitting	✓		✓
Fitting complexity	high	medium	low
Fitting acc. CDF smooth	high	high	high
Fitting acc. CDF peaky	high	n/a	medium
Fitting accuracy ACF	high	n/a	very high
Fitting accuracy XCF	n/a	n/a	high
Sample generation complexity	low, $\mathcal{O}(N)$	lower, $\mathcal{O}(N)$	lowest, $\mathcal{O}(N)$

TABLE 5.1: Comparison of generic traffic models.

(ii) TES models and (iii) transformed Gaussian ARMA models; they are summarized below and a respective comparison is given in Table 5.1, cf. [178].

5.1.1.1 Markovian Models

This type of models is most often encountered in literature [179] [180] within various different contexts, such as speech [181], video [162] and online gaming [182]. It comes in various flavors, for example, MMPP or Markovian Arrival Process (MAP). They base on a hidden Markov chain generating state dependent arrivals, which are summarized to a common random process. This yields a highly flexible structure, which is able to characterize arbitrary CDFs and ACFs jointly. Furthermore, the resulting processes are fully analytically tractable. The drawback of this approach appears when the model is fitted to data; namely, the CDF and ACF have to be fitted jointly. This implies that (i) the fitting process has to iterate between CDF and ACF, which is computationally intensive, and (ii) the number of parameters to be fitted for both CDF and ACF is coupled, which is not optimal from a parsimoniousness point of view. Recent work in the field tackles this problem and achieves good fitting performance with a low amount of model parameters [180]. Further, MAPs have been extended to capture the XCF of multiple processes in [179], where the authors accurately fit their model to various traffic types.

5.1.1.2 TES Models

The acronym TES stands for *Transform Expand Sample*, an approach based on uniform random processes [183] [184]. The ACF and CDF are introduced to the process in two steps which are decoupled. This is due to the fact that auto-correlations can be introduced to an uniform random process without changing its distribution. The model benefits from the vast amount of available transformations from uniform distributions to any other type of distribution, which is the basis of all random numbers in computers [165]. Nevertheless, the method faces problems with the smoothness of the sample paths and with fitting auto-correlations at large lags, which requires the intervention of the user during the fitting procedure.

5.1.1.3 Transformed Gaussian ARMA Models

The last category, TARMA models, comprises various works from different fields of study [185] [186] [187] [163] [188]. It bases on correlated Gaussian random processes which are warped by a memoryless non-linear transformation. The ACF and CDF are introduced in two decoupled steps, first the ACF, depolying regression models (e.g., ARMA models), then the CDF by a non-linearity. As both other approaches, transformed Gaussian models in its general form are able to reproduce any desired CDF. Further, also a vast range of ACFs are captured, which can be fitted parsimoniously due to the evolved methods of linear system theory [166] [177] [164]. The ACF of the output process is further analytically tractable, which is in general computationally expensive. In order to overcome this problem, Hermitian polynomials are proposed as non-linearity [185], for which closed form solutions can be found for ACF and XCF after transformation (cf. [186, pp. 419–426] [189, pp. 132ff.]). This approach is further chosen for the work presented in this section due to its flexibility.
In the context of Internet traffic modeling, Auto-Regressive To Anything (ARTA) models have to be mentioned [187]. The authors show that the approach is suited for traffic modeling and provide a general analytical framework, by leaving the non-linearity unspecified. Recent work in the field of ARTA modeling extends this method to a combination of ARMA models with Markov models [190]. The authors show that ARMA processes are suitable for introducing correlations into phase-type distributions. Similar approaches appeared in video traffic modeling [155] [163], where the focus is mainly on the generation of correlated chi-squared and gamma processes. The method is referred to as Gaussian Auto-Regressive and Chi-Squared (GACS) models and is fully analytically tractable.
The generation of Gaussian random processes with cross-correlation is preferable over other methods because of the numerously available literature (cf. [166, pp. 551ff.] [177, pp. 401ff.]) and the possibility of decoupling the fitting problem of ACFs and XCF. A theoretical framework for the generation of multiple random processes based on ARTA models is given in [191]. In the field of video modeling, the modeling of correlation coefficients (i.e., XCFs at lag 0) is treated in [192] [193]; the results show an improvement over models which neglect cross-correlations. A recently presented method [194] additionally allows to fit multidimensional joint distribution functions of multiple random processes.

5.1.2 Generating Traffic From Transformed ARMA Models

In this section I explain the functional principle of the TARMA modeling approach. Further, the generation/emulation of data traffic from given model parameters is described, which may act as input for network simulations. For the rest of this section the physical quantities of the output processes $Z_i[n]$ are not specified; however, common examples are PS and IAT.
The proposed modeling approach allows for the joint representation of arbitrary CDFs, ACFs and XCFs. This is achieved by three sequential transformations of normal i.i.d. random processes, each of which being responsible for handling one of the above statistical measures. A corresponding block diagram for the generation of I output processes $Z_i[n]$ is depicted in Figure 5.2. The four different types of blocks have the following functionalities:

1 **Gaussian i.i.d. random process**. These blocks generate J independent normal random processes with zero mean and unit variance.

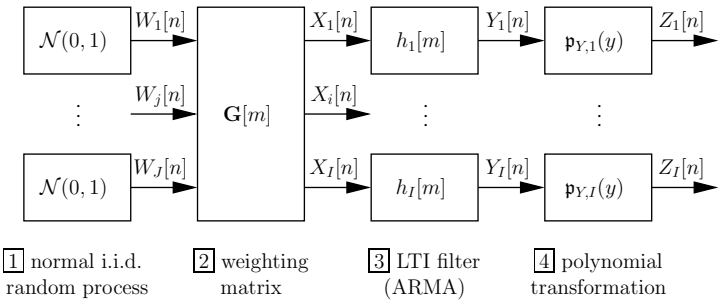


FIGURE 5.2: Block diagram of the proposed modeling approach for the generation of network source traffic, I inter-dependent random processes.

2 **Weighting matrix**. Matrix $\mathbf{G}[m]$ introduces cross-correlation to the I output processes $X_i[n]$.

3 **LTI filter (ARMA)**. The Linear Time Invariant (LTI) filters $h_i[m]$ introduce auto-correlations to the processes $Y_i[n]$.

4 **Polynomial transformation**. The memoryless polynomials $\mathfrak{p}_{Y,i}(y)$ shape the distributions of $Z_i[n]$.

Between the four blocks, the intermediate random processes $W_j[n]$, $X_i[n]$ and $Y_i[n]$ are observed; they can be interpreted as the passing of one random sample per time index n from one block to its successor. For ensuring that the mentioned blocks fulfill their task properly, the following restrictions have to be satisfied for the respective input processes:

1. The CDF of the process is normal (Gaussian) with zero mean and unit variance,
2. The ACF of the process is zero for all lags $m \neq 0$,
3. The XCF between the processes is zero.

For the process $W_j[n]$ all three conditions must apply, for $X_i[n]$ the first two conditions are necessary, leading to Eq. (5.14) and Eq. (5.15), and for $Y_i[n]$ only the first condition is required, leading to Eq. (5.7). Sticking to those requirements also guarantees that the fitting problems for CDF, ACF and XCF are separable, one of the key features of the proposed model.
In the following each functional block is described in detail. For convenience, a summary of the rest of this section is anticipated:

1. Samples of $Z_i[n]$ are generated by the consecutive accomplishment of Eq. (5.12), Eq. (5.6) and Eq. (5.1) on normal i.i.d. samples.
2. The Probability Density Function (PDF) of $Z_i[n]$ can be calculated by Eq. (5.3).
3. ACFs can be calculated by Eq. (5.9) and Eq. (5.4).
4. XCFs can be assessed by Eq. (5.13), Eq. (5.11) and Eq. (5.5).

This framework allows for the complete analytical tractability of the processes $Z_i[n]$. The description of the functional blocks of Figure 5.2 is given in the following in reverse order. If used in a unique manner, the index i of the random processes will be dropped.

5.1.2.1 Memoryless Polynomial Transformation

The last block of the chain, block 4, performs a polynomial transformation $\mathfrak{p}_Y(y)$ of the random process $Y[n]$, according to

$$Z[n] = \mathfrak{p}_Y(Y[n]) = \sum_{p=0}^{P} \alpha_p \cdot (Y[n])^p, \tag{5.1}$$

where P is the order of the polynomial $\mathfrak{p}_Y(y)$ and α_p are the coefficients of the polynomial for the power p. The goal of this transformation is to resemble a quantile transformation procedure. It enables the generation of a random variable $Z[n]$ with arbitrary distribution, from any other distribution of $Y[n]$ by mapping the corresponding percentiles to each other [164, p. 139]. This is according to

$$Z[n] = F_Z^{-1}(F_Y(Y[n])), \tag{5.2}$$

where $F_Y(\cdot)$ denotes the CDF of the random process $Y[n]$ and $F_Z^{-1}(\cdot)$ the inverse of the desired CDF of the therewith created random process $Z[n]$.
In the present case $F_Y(\cdot)$ is a Gaussian CDF (i.e., complementary Q-function), since $Y[n]$ is normal distributed with zero mean and unit variance, which is guaranteed by the restrictions imposed on $Y[n]$ mentioned above. The polynomial $\mathfrak{p}_Y(y)$ shall resemble this percentile transformation procedure, $\mathfrak{p}_Y(y) \approx F_{Z,\text{target}}^{-1}(F_Y(y))$, thus, define the targeted distribution for the output process $Z[n]$. Proximity of the targeted CDF $F_{Z,\text{target}}(\cdot)$ and the actually realized CDF $F_Z(\cdot)$ is achieved by an ordinary polynomial curve fitting, for example, with the least-squares method [195]. Furthermore, Section 5.1.4 shows that for various types of distributions $F_{Z,\text{target}}(\cdot)$, a low-order polynomial approximation $p_Y(y)$ is satisfactory.
The exact PDF $f_Z(z)$ of the output process $Z[n]$ can be computed according to Appendix E, Corollary E.1, by

$$f_Z(z) = \mathsf{f}(\alpha_p, y_k(z)), \tag{5.3}$$

where $\mathsf{f}(\cdot)$ denotes a function specified in Corollary E.1, and $y_k(z)$ denote the real roots of the equation $\mathfrak{p}_Y(y) = z$.
The input process $Y[n]$ exhibits non-trivial auto-correlations and cross-correlations, introduced by the preceding blocks (i.e., 2, 3). Those are influenced by the polynomial transformation. The auto-correlation function $\rho_{ZZ}[m]$ of the output process $Z[n]$ is a transformed version of the auto-correlation function $\rho_{YY}[m]$. In Appendix E, Corollary E.8, it is shown that

$$\rho_{ZZ}[m] = \mathfrak{p}_\rho(\rho_{YY}[m]), \tag{5.4}$$

where $\mathfrak{p}_\rho(\rho)$ is a polynomial, depending on the coefficients α_p of the polynomial $\mathfrak{p}_Y(y)$. Similarly, the cross-correlation function $\rho_{Z_i Z_l}[m]$ between the processes $Z_i[n]$ and $Z_l[n]$ is a transformed version of the XCF $\rho_{Y_i Y_l}[m]$ between the processes $Y_i[n]$ and $Y_l[n]$. In Appendix E, Corollary E.9, it is shown that

$$\rho_{Z_i Z_l}[m] = \mathfrak{p}_{\rho,il}(\rho_{Y_i Y_l}[m]), \tag{5.5}$$

where $\mathfrak{p}_{\rho,il}(\rho)$ is a polynomial, depending on the coefficients $\alpha_{p,i}$ of the polynomial $\mathfrak{p}_{Y,i}(y)$ and $\alpha_{p,l}$ of the polynomial $\mathfrak{p}_{Y,l}(y)$.

5.1.2.2 Linear Time Invariant Filter

The process $X[n]$ is passed through an LTI filter with real-valued impulse response $h[m]$, block $\boxed{3}$. This filter fulfills the task of introducing auto-correlation to $X[n]$, resulting in the process

$$Y[n] = \sum_{m=-\infty}^{\infty} X[m] \cdot h[n-m] = X[n] * h[n], \tag{5.6}$$

where $*$ denotes the convolution operation. The reason for the combination of a Gaussian process with an LTI filter is the closure property of the set of all Gaussian processes on the addition operation and, especially, on linear combinations. It implies that any Gaussian random process $X[n]$ which is transformed by a linear filter $h[m]$ results again in a Gaussian process $Y[n]$. Thereby, the mean and variance of the output process are changed to [164, p. 398], $\mu_Y = \sum_{m=-\infty}^{\infty} h[m] \cdot \mu_X$ and $\sigma_Y^2 = \sum_{m=-\infty}^{\infty} (h[m])^2 \cdot \sigma_X^2$, where μ denotes the mean and σ^2 the variance. If the Gaussian input sequence $X[n]$ has zero mean and unit variance (which is one of the requirements mentioned above) and the sum of all squared filter coefficients $h[m]$ equals one, it is guaranteed that the distribution of the output sequence $Y[n]$ is also Gaussian with zero mean and unit variance and fulfills the restrictions on $Y[n]$. Hence, the closure property allows to introduce an ACF to the random process $X[n]$ by an arbitrary linear filter $h[m]$ without changing its distribution, provided it satisfies

$$\sigma_h^2 = \sum_{m=-\infty}^{\infty} (h[m])^2 \stackrel{!}{=} 1. \tag{5.7}$$

The *auto-correlation function* for a (wide-sense) stationary identically distributed random process $Y[n]$ is thereby defined as

$$\begin{aligned} \rho_{YY}[m] &\doteq \frac{\gamma_{YY}[m] - \mu_Y^2}{\sigma_Y^2} \\ &= \frac{E\{(Y[n] - \mu_Y)(Y[n+m] - \mu_Y)\}}{\sigma_Y^2}, \end{aligned} \tag{5.8}$$

with the expectation operation $E\{\cdot\}$, the mean μ_Y, the variance σ_Y^2 and the *unnormalized* ACF $\gamma_{YY}[m]$. The term *cross-correlation function* is defined similar, by exchanging the random process $Y[n]$ with two distinct processes $Y_i[n]$ and $Y_l[n]$, yielding $\rho_{Yi,Yl}[m]$. The ACF introduced by the LTI filter $h[m]$ to the process $Y[n]$ calculates to (cf. [164, p. 401])

$$\begin{aligned} \rho_{YY}[m] &= \frac{\gamma_{YY}[m]}{\sigma_Y^2} = \frac{\gamma_{hh}[m] * \gamma_{XX}[m]}{\sigma_h^2 \cdot \sigma_X^2} \\ &= \frac{\gamma_{hh}[m] * \delta[m]}{\sigma_h^2 \cdot 1} = \frac{\sigma_h^2 \cdot \rho_{hh}[m]}{\sigma_h^2} \\ &= \rho_{hh}[m] = h[m] * h[-m]. \end{aligned} \tag{5.9}$$

with the unit impulse sequence $\delta[m]$. The condition in Eq. (5.7) does not effect the auto-correlation function $\rho_{YY}[m]$, since it is normalized by the variance of the output process σ_h^2, as observed in the above equation. Therefore, any scaled version of the applied LTI filter results in the same autocorrelation function. Conversely, this means that Eq. (5.7) can always be satisfied by scaling any arbitrary $h[m]$ with a constant. This fact **decouples the problems of fitting a CDF and an ACF** to data, being important for a parsimonious and efficient treatment of the overall fitting problem (which is the main reason for the choice of this model).

The linear filter is composed of two components, an Auto-Regressive (AR) component $\phi(B)$ and a Moving-Average (MA) component $\theta(B)$, which together constitute the ARMA model. The AR element feeds a linear combination of the past output values $Y[n-m]$ back to the actual output value, the MA unit feeds a linear combination of the past input values $X[n-m]$ to the actual output $Y[n]$. By introducing the backshift operator B (i.e., $B\,X[n]=X[n-1]$), $\phi(B)$ and $\theta(B)$ can be interpreted as polynomials in B, where the power of B indicates how often a backshift is performed. The linear filter satisfies the difference equation (cf. [166, pp. 8ff.])

$$\phi(B) \cdot Y[n] = \theta(B) \cdot X[n]. \tag{5.10}$$

Assessing the system behavior relies on the calculation of the impulse response $h[m]$ from the ARMA parameters $\phi(B)$ and $\theta(B)$. This can be achieved recursively by assuming $X[n]=\delta[n]$ and $h[n]=Y[n]$, starting from index $n=0$ and approaching $n=\infty$. Besides, solving the difference equation for $Y[n]$, results in the polynomial $\psi(B)=\phi^{-1}(B)\theta(B) = \sum_{m=0}^{\infty}\psi_m B^m$, which directly leads to the impulse response by assigning $h[m]\doteq\psi_m$, $\forall\, 0\leq m\leq\infty$.
The linear filters $h_i[m]$ affect the cross-correlation function $\rho_{Y_iY_l}[m]$ between $Y_i[n]$ and $Y_l[n]$. In analogy to Eq. (5.9) we obtain

$$\begin{aligned}\rho_{Y_iY_l}[m] &= \rho_{X_iX_l}[m] * \rho_{h_ih_l}[m] \\ &= \rho_{X_iX_l}[m] * h_i[m] * h_l[-m].\end{aligned} \tag{5.11}$$

This equation allows for the analytical calculation of the transformation of the XCF induced by the introduction of ACFs to the random processes. Hence, alike Eq. (5.4) and Eq. (5.5), this equation is the key feature for the separation of the fitting problems of ACFs and XCFs.

5.1.2.3 Weighting Matrix

Weighting matrix $\mathbf{G}[m]$, block $\boxed{2}$, serves the purpose of introducing cross-correlations into the processes $X_i[n]$. It combines the processes $W_j[n]$ by weighted addition. However, the elements $g_{ij}[m]$ of matrix $\mathbf{G}[m]$ are sequences of weights in the timing lag m, in the most general case. This allows for the interpretation of each $g_{ij}[m]$ as the impulse response of a linear filter or, equivalently, as polynomial $g_{ij}(B)$ in the backshift operator B. Thus, matrix $\mathbf{G}[m]$ is equivalent to a matrix polynomial $\mathbf{G}(B)$ in B (cf. [166, pp. 551ff.] [177, pp. 401ff.]). The input-output relation can be conveniently written in matrix notation as

$$\mathbf{X}[n] = \mathbf{G}(B) \cdot \mathbf{W}[n], \tag{5.12a}$$

where $\mathbf{X}[n]$ and $\mathbf{W}[n]$ are vector valued random processes composed by all $X_i[n]$ and $W_j[n]$. On the other hand, the element-wise output relation can be written as

$$X_i[n] = \sum_{j=1}^{J} g_{ij}[n] * W_j[n], \tag{5.12b}$$

in which each element $g_{ij}[m]$ denotes a linear filter in m.
The cross-correlations introduced by matrix $\mathbf{G}[m]$ can be calculated by deploying the backshift notation $\mathbf{G}(B)$, namely,

$$\mathbf{\Gamma}_X(B) = \mathbf{G}(B) \cdot \mathbf{G}^T(B^{-1}), \tag{5.13a}$$

where $(\cdot)^T$ denotes the transposed of the matrix. The corresponding matrix in the *time lag* domain is denoted by $\mathbf{\Gamma}_X[m]$, with each element $\gamma_{X_iX_l}[m]$ being the specific XCF between the respective random processes $X_i[n]$ and $X_l[n]$. These elements calculate to

$$\rho_{X_iX_l}[m] = \gamma_{X_iX_l}[m] = \sum_{j=1}^{J} g_{ij}[m] * g_{lj}[-m], \tag{5.13b}$$

where $\gamma_{X_iX_l}[m]$=$\rho_{X_iX_l}[m]$ due to the normalization postulated by Eq. (5.14).
As already mentioned, $\mathbf{G}[m]$ is restricted to the set of matrices which fulfill the following conditions for the output processes: (i) all $X_i[n]$ must be Gaussian distributed with zero mean and unit variance and (ii) all $X_i[n]$ must have zero auto-correlation for lags $m\neq0$. The first condition requires that the squared sum of all row elements $g_{ij}[m]$ of $\mathbf{G}[m]$ equals one for all rows i,

$$\sum_{j=1}^{J} \sum_{m=-\infty}^{\infty} (g_{ij}[m])^2 \stackrel{!}{=} 1. \tag{5.14}$$

Due to the closure property of the Gaussian distribution on linear combinations, Gaussianity as well as the zero mean are preserved for $X_i[n]$. A squared sum equal to one, cf. Eq. (5.14), ensures that the variance of $X_i[n]$ equals one; hence, the first condition is fulfilled.
The second condition (i.e., zero ACF for all lags unequal to zero) is equivalent to forcing all diagonal elements of $\mathbf{\Gamma}_X[m]$ to

$$\rho_{X_iX_i}[m] \stackrel{!}{=} 1 \cdot B^0 = 1. \tag{5.15}$$

It must be ensured by the respective fitting procedure (cf. Section 5.1.3.3). The condition guarantees that the ACFs of the output processes are independent of the matrix $\mathbf{G}[m]$. This seems to be an overhead, since $\mathbf{G}[m]$ could also introduce an ACF to the processes and, thereby, incorporate the linear filter $h[m]$. The reason of the separation of the two blocks is the possibility of different targeted fitting accuracies for ACF and XCF. Specific types of data traffic may, for example, require that the ACF is modeled accurately up to a lag of 10^4, whereas it is considered as sufficient to model the XCF only at lag 0. This task is simplified by splitting both fitting problems into two independent sub-problems. Further, fitted models tend to have less parameters in this case.
An important class of matrices which satisfies this condition is the set of real valued matrices $\mathbf{G}$ without any backshift operation. Such matrices only define the cross-correlation coefficients $\rho_{X_iX_l}[0]$ between the processes $X_i[n]$ and $X_l[n]$ and do not introduce cross-correlations at any other lag. This is sufficient for many practical applications (cf. Section 5.1.4.1). In this case matrix $\mathbf{G}(B)$≡$\mathbf{G}[m]$≡$\mathbf{G}$ does not contain memory and the input-output relation Eq. (5.12) reduces to an ordinary matrix multiplication $\mathbf{X}[n]$=$\mathbf{GW}[n]$. The number J of processes $W_j[n]$ may be less or equal to the number I of output processes $Z_i[n]$, $J\leq I$, in order to achieve all possible combinations of correlation coefficients; see Section 5.1.3.3 for details. The analytical calculation of the cross-correlations reduces to the ordinary matrix multiplication $\mathbf{\Gamma}_X$=$\mathbf{GG}^T$ in this case.

5.1.2.4 Normal i.i.d. Processes

The proposed traffic generation method requires J i.i.d. Gaussian random processes $W_j[n]$ with zero mean and unit variance, block $\boxed{1}$. This is convenient for simulation

purposes, since most modern computer systems provide predefined, computationally efficient routines for the generation of high-quality normal distributed random variables [165]. Furthermore, all J processes must be independent, hence, can be interpreted as a J-dimensional i.i.d. random process. The generation of such processes is feasible up to high dimensionality [196]. Thus, all three requirements for the intermediate process $W_j[n]$ imposed at the beginning of this section are fulfilled. The number of processes J is determined by the number of output processes I and the desired structure of interdependencies (i.e., XCFs). The exact number is determined during the fitting process, see Section 5.1.3.

5.1.3 Building TARMA Models for Recorded Traffic

Procedures for fitting TARMA models to measurement data are illustrated in the following. Thereby, well established methods for fitting ARMA processes and polynomial regression are partly reused. Scripts performing the fitting process fully automatic can be downloaded at [121]. The fitting process has to be performed beginning with the polynomial and proceeding in reverse order, from block $\boxed{4}$ to $\boxed{2}$, cf. Figure 5.2. By doing so the fitting problems for each block are decoupled. Nevertheless, the fitting procedure for each component has to account for the influences of the consecutive components on the respective statistical measure. For example, the ACF which is introduced by the linear filter $h[m]$ is altered by the polynomial $\mathfrak{p}_Y(y)$. The influences can be assessed analytically by the functions presented in Section 5.1.2, e.g., Eq. (5.4), Eq. (5.5) and Eq. (5.11). This is one of the big advantages of the proposed model. In the following I describe the fitting processes for each block separately. Thereby the targeted quantities of the resulting model are denoted by the subscript $(\cdot)_{\text{target}}$; those are, for example, obtained from measured data traffic or from analytical models.

5.1.3.1 The Polynomial

The first block to be considered is the polynomial transformation $\mathfrak{p}_Y(\cdot)$, block $\boxed{4}$. It shall introduce an arbitrary CDF to $Z[n]$. As already stated in Section 5.1.2.1, $\mathfrak{p}_Y(\cdot)$ shall approximate a quantile-transformation procedure, confer Eq. (5.2). This can be achieved by solving a least-squares fitting problem [195]. Thereby the sample points to be fit by polynomial regression are pairs of ω_k-quantiles $(\Omega_Z, \Omega_Y)_k$ from the Gaussian CDF $F_Y(\cdot)$ of $Y[n]$ and the targeted CDF $F_{Z,\text{target}}(\cdot)$ of $Z[n]$, namely,

$$(\Omega_Z, \Omega_Y)_k = \left(F^{-1}_{Z,\text{target}}(\omega_k), F^{-1}_Y(\omega_k)\right) \qquad 0 \leq \omega_k \leq 1.$$

The sample points $(\Omega_Z, \Omega_Y)_k$ can be arranged in a Q-Q-plot. An illustration of the determination of sample points is given in Figure 5.3, wherein the process $Z[n]$ has uniform distribution. The number of quantile values ω_k for the polynomial regression, as well as their position and spacing is an open point for optimization; hence, depending on the designers needs. For the rest of this work equidistant spacing from zero to one is assumed, $0 \leq \omega_k \leq 1$, excluding both limiting values, since they would yield (Ω_Z, Ω_Y)=$(\Omega_Z, \pm\infty)$ and are not suited for a polynomial regression. It is irrelevant if the quantile values either stem from measurements or a certain type of analytical distribution.

The order P of the polynomial plays an important role for the quality of the fit. If the order is high enough, it is possible to fit any number of points with arbitrary accuracy; however, polynomial fitting has poor interpolation properties, i.e., the fit tends to oscillate between points (over-fitting). I recommend to use low-order polynomials.

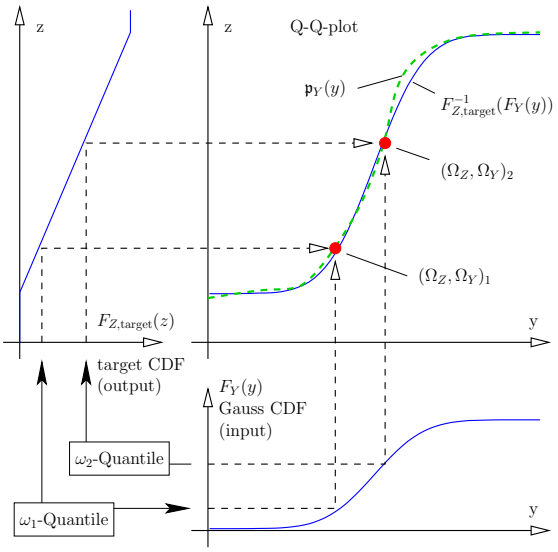


FIGURE 5.3: Q-Q-plot for obtaining the sample points $(\Omega_Z, \Omega_Y)_k$ to which the polynomial $\mathfrak{p}_Y(y)$ must be fitted.

Furthermore, the computational complexity for the generation of random samples is strongly reduced for small P. A comparison of the quality-of-fit for different polynomial orders is given in Section 5.1.4.

5.1.3.2 The Linear Filter

The next block to be considered is block 3, the linear filter $h[m]$. This block shall introduce an ACF to $Z[n]$ by introducing a respective ACF to $Y[n]$. In literature two approaches for ARMA modeling are prevalent: (i) ACF based approaches (e.g., Yule-Walker equations, Power Spectral Density (PSD) based approaches), which require $\rho_{YY,\text{target}}[m]$ as input and (ii) direct methods based on the data itself (e.g., Maximum Likelihood (ML) modeling), requiring $Y_{\text{target}}[n]$ as input. It has to be taken into account that the polynomial transformation $\mathfrak{p}_Y(y)$, block 4, influences the ACF of the output process $Z[n]$ according to Eq. (5.4). According to which fitting method shall be applied, the respective input quantity has to be pre-distorted such that the influence of the polynomial transformation is taken into account. A respective graphical representation is given in Figure 5.4. Thus, either (i) the input data for fitting has to be manipulated, according to $Y_{\text{target}}[n]{=}\mathfrak{p}_Y^{-1}(Z_{\text{target}}[n])$ (cf. Figure 5.4: Step 1A) or (ii) the input ACF for fitting has to be manipulated, according to $\rho_{YY,\text{target}}[m]{=}\mathfrak{p}_\rho^{-1}(\rho_{ZZ,\text{target}}[m])$ (cf. Figure 5.4: Step 2B).
Both of these pre-distortion methods require the inversion of a polynomial, or, equivalently, finding the respective roots, which is analytically not feasible for any $P{>}4$. However, this frequent problem can efficiently be solved numerically with high accuracy. The polynomial $\mathfrak{p}_\rho^{-1}(\cdot)$ is usually smoother than the polynomial $\mathfrak{p}_Y^{-1}(\cdot)$ and allows for a unique solution of the inversion problem. Therefore, the fitting procedure involving the transformation of the ACF (cf. Figure 5.4: Step 2A–2B–2C) yields most probably better results than the direct fitting approach involving the transformation of the data (cf. Figure 5.4: Step 1A–1B or 1A–3B–2C). A counterexample is modeling of video sequences, see Section 5.1.4.2.
Having obtained either $\rho_{YY,\text{target}}[m]$ or $Y_{\text{target}}[n]$ as input for the respective fitting procedure for the linear filter, any arbitrary ARMA modeling approach can be applied for

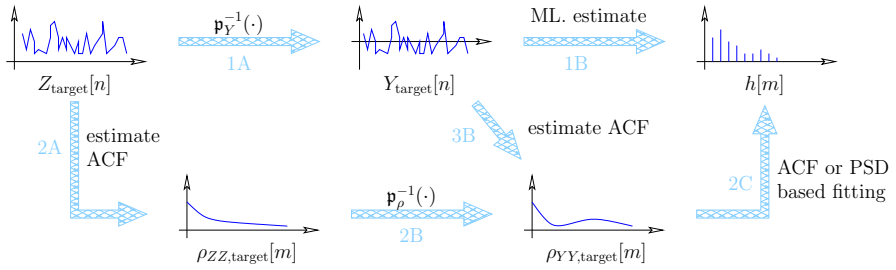


FIGURE 5.4: Possible approaches to fit a linear filter (ARMA model) to traced data.

computing the ARMA parameters $\phi(B)$ and $\theta(B)$. Those are numerous in literature [166] [177], including various software solutions.
A typical property of ACFs of network traffic is Long Range Dependence (LRD). It is encountered for time series of various quantities of data traffic, such as packet-sizes, flow durations, packet counts and packet inter-arrival times [197]. LRDs have an impact on the queueing performance and are therefore important to be captured. However, several ARMA modeling procedures have problems to capture these effects. In Appendix F, a method is presented which overcomes these problems: a variant of ACF-based fitting. It yields a parsimonious ARMA model with finite length and is thus only an approximation to an LRD process. Nevertheless, the fitting accuracy is high for any finite lag of the ACF and the generation of samples exhibits very low complexity. Prominent alternatives are the well-known Auto-Regressive Fractionally Integrated Moving-Average (ARFIMA) models [166, pp. 428ff.] [198], yielding long-range dependent processes by fractional integration (summation). This can be translated to an equivalent ARMA model of (formally) infinite length, for which the traffic synthesis is computationally more expensive than for ordinary ARMA processes.
Finally, in order to suffice the restriction on $Y[n]$ (i.e., normally distributed with zero mean and unit variance), it has to be guaranteed that Eq. (5.7) is satisfied. This can readily be achieved by scaling $\theta(B)$ with an appropriate constant.

5.1.3.3 The Weighting Matrix

The last block to be considered for the fitting procedure is block $\boxed{2}$, the weighting matrix $\mathbf{G}[m]$. This block shall introduce XCFs between the I different output processes $Z_i[n]$, by introducing respective XCFs to $X_i[n]$. Again the influences from block $\boxed{3}$ and $\boxed{4}$ on the XCFs between the output processes have to be considered first; confer Eq. (5.11) and Eq. (5.5). In analogy to the fitting problem for the linear filter it is possible to either (i) fit $\mathbf{G}[m]$ to the random processes $X_{i,\text{target}}[n]$ or (ii) fit $\mathbf{G}[m]$ to all the XCFs $\rho_{X_iX_l,\text{target}}[m]$. How to obtain one of the above quantities is outlined in Figure 5.5.
The first procedure is accomplished by (cf. Figure 5.5: 1A–1B–1C): (i) inverting the polynomial transformation, $Y_{i,\text{target}}[n]{=}\mathfrak{p}_Y^{-1}(Z_{i,\text{target}})$, (ii) whitening the obtained sequence by applying $X_{i,\text{target}}[n]{=}Y_{i,\text{target}}[n]{*}h_i^{-1}[n]$ and (iii) applying a Maximum Likelihood (ML) fitting approach.
The second procedure comes in several flavors (cf. Figure 5.5). For example, option 2A–2B–2C–2D can be performed as follows: (i) calculate the XCFs of $Z[n]$, (ii) apply the inverse polynomial from Eq. (5.5) to it, $\rho_{Y_iY_l,\text{target}}[m]{=}\mathfrak{p}_{\rho,il}^{-1}(\rho_{Z_iZ_l,\text{target}}[m])$, (iii) whiten the XCFs of $Y[n]$ by the inverse filters $h_i^{-1}[m]$ of Eq. (5.11), $\rho_{X_iX_l,\text{target}}[m]{=}\rho_{Y_iY_l,\text{target}}[m] * h_i^{-1}[m]{*}h_l^{-1}[-m]$ and (iv) calculate $\mathbf{G}[m]$ from Eq. (5.13) using the *Cholesky decomposition*.

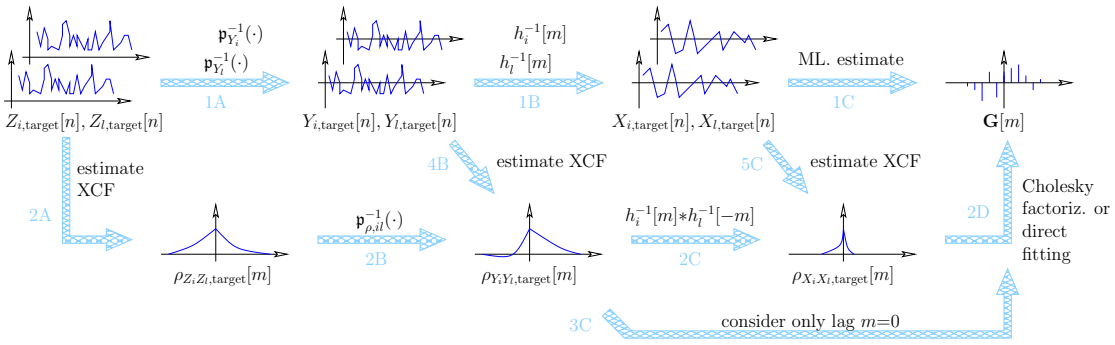


FIGURE 5.5: Possible approaches to fit a polynomial matrix $\mathbf{G}[m]$ to traced data.

As already mentioned in the context of ACF modeling, step 1A for inversion of the polynomials $\mathfrak{p}_{Y_i}^{-1}(\cdot)$ is problematic and shall be avoided. Fitting sequences which include it (e.g., 1A–1B–1C, cf. Figure 5.5) are therefore unfavorable and yield alternative sequences (e.g., 2A–2B–2C–2D or 2A–2B–3C, cf. Figure 5.5) more convenient for practical use.

Consequently, the central fitting step corresponds to either Step 2D or Step 3C. Two methods are available for this purpose: (i) the Cholesky-factorization or (ii) the direct method (only applicable to the case of two output processes). Unfortunately, both methods are not parsimonious (in contrast to the modeling approaches for PDFs and ACFs). They yield roughly one model parameter per lag for each XCF; consequently, the number of parameters becomes easily prohibitively large. Therefore I suggest to model only up to a few lags of the XCF (e.g., only lag 0), especially if more than two output processes shall be characterized. An evaluation of the impact of the number of considered lags on the model accuracy is given in Section 5.1.4.1. The two fitting algorithms are described in the following paragraphs.

Cholesky-factorization: For fitting $\mathbf{G}(B)$ by a Cholesky decomposition, an auxiliary process of random vectors $\mathbf{X}'[n]$ has to be constructed. It has $(2M+1)I$ dimensions, where M denotes the maximum lag to be modeled and I the number of output processes I. It consists of $2M+1$ shifted versions of each output process $X_i[n-m]$, with $m=-M,\ldots,M$ and $i=1,\ldots,I$, arranged according to

$$\mathbf{X}'[n] = (X_1[n-M],\ldots,X_1[n+M],X_2[n-M],\ldots,X_I[n-M],\ldots,X_I[n+M])^T. \quad (5.16)$$

The correlation matrix $\mathbf{\Gamma}'_X$ of $\mathbf{X}'[n]$ has to be constructed, whereof the single elements are all coefficients of the auto and cross-correlation functions of $X_i[n]$. Notice, that Eq. (5.15) must be satisfied; hence, $\mathbf{\Gamma}'_X$ is a block matrix with identity matrices on the diagonal.

Performing the Cholesky decomposition of $\mathbf{\Gamma}'_X$ yields a lower triangular matrix, which has to be normalized by $\frac{1}{\sqrt{2M+1}}$ in order to satisfy Eq. (5.14). Each column of this matrix can be divided into I blocks of length $2M+1$; thereby, each block is translated to one element $g_{ij}[m]$ of $\mathbf{G}[m]$, whereas each element within a block is equivalent to a coefficient of $g_{ij}[m]$ at a specific lag m with $m=-M,\ldots,M$.

Consequently, the number J of required input processes $W_j[n]$ amounts to $J=(2M+1)I$; the number of model parameters (i.e., sum of all non-zero coefficients $g_{ij}[m]$) calculates to $\frac{1}{2}\,J\,(J-1)$.

Direct Fitting: If the number of output processes is $I=2$, then it is possible to pursue a direct fitting approach. It is usually more economic than the Cholesky decomposition,

both in the number of input processes $Wj[n]$ (amounting to $J{=}2M{+}2$) and the model parameters (i.e., $2J$).
It is based on the observation that Eq. (5.15) (i.e., zero ACF for all lags unequal to zero) can be satisfied by forcing each of the polynomials $g_{ij}(B)$, to monomials in the backshift operator B

$$g_{ij}(B) \stackrel{!}{=} g_{ij,l} \cdot B^l \tag{5.17}$$

where $g_{ij,l}$ denotes the only non-zero element of $g_{ij}[l]$ located at lag $m{=}l$. The monomial order l may vary from element to element. Thus, each element $g_{ij}[m]$ is a moving-average filter with only one timing lag. This ensures that each sample of each process $W_j[n]$ appears only once within all the samples of the process $X_i[n]$ and, consequently, does not cause any auto-correlations in $X_i[n]$.
Accordingly, the polynomial matrix $\mathbf{G}(B)$ shall be constructed as

$$\begin{aligned} c &= \sum_{m=-M}^{M} |\rho_{X_1X_2}[m]| \\ g_{1,j}(B) &= \operatorname{sign}(\rho_{X_1X_2}[j{-}M{-}1])\ \sqrt{c\ |\rho_{X_1X_2}[j{-}M{-}1]|} \cdot B^0 \\ g_{2,j}(B) &= \sqrt{\tfrac{1}{c}\ |\rho_{X_1X_2}[j{-}M{-}1]|} \cdot B^{j-M-1} \\ \mathbf{G}(B) &= \begin{pmatrix} g_{1,1}(B) & \cdots & g_{1,2M+1}(B) & \sqrt{1-c^2}\ B^0 \\ g_{2,1}(B) & \cdots & g_{2,2M+1}(B) & 0 \end{pmatrix}, \end{aligned} \tag{5.18}$$

whereas $\operatorname{sign}(\cdot)$ denotes the sign operator. In this case the targeted XCF $\rho_{X_1X_2}[m]$ is induced to the processes, without causing any auto-correlations.
Considering only lag zero: Both methods, the Cholesky-decomposition and the direct fitting, converge if the maximum lag $M{=}0$ and $I{=}2$ output processes $X_i[n]$ with $\rho_{X_1X_2}[0]{=}g_{21}$. Then the single parameter g_{21} is enough to specify the matrix $\mathbf{G}(B)$ by

$$\mathbf{G} = \begin{pmatrix} g_{21} & \sqrt{1-g_{21}^2} \\ 1 & 0 \end{pmatrix}, \qquad \text{yielding} \qquad \mathbf{\Gamma} = \begin{pmatrix} 1 & g_{21} \\ g_{21} & 1 \end{pmatrix}. \tag{5.19}$$

The restrictions on $X_i[n]$ are inherently satisfied by this fitting approach: (i) Eq. (5.17) is satisfied since only $g_{il}[0]$ is considered for any two processes $X_i[n]$ and $X_l[n]$. (ii) Moreover, Eq. (5.14) is satisfied for $\mathbf{G}$, since the cross-correlation matrix $\mathbf{\Gamma}$ has unit diagonal elements.
In order to determine the value of g_{21}, it is not required to perform the whitening procedure (cf. Figure 5.5: Step 2C), since the correlations introduced by $h_i[m]$ equal one at lag zero. Instead, the cross-correlation coefficient $\rho_{Y_iY_l,\text{target}}[0]$ can be directly equated with g_{21} (cf. Figure 5.5: Step 3C),

$$g_{21} = \rho_{Y_iY_l,\text{target}}[0]. \tag{5.20}$$

This yields usually better results than including the whitening procedure, since inaccuracies introduced by the model of the ACFs, are suppressed. Considering only lag zero is a parsimonious way of resembling XCFs and, thus, recommended for traffic modeling purposes. A comparison of the different fitting strategies is provided in Table 5.2.

5.1.4 Evaluation

It remains to evaluate the proposed modeling approach and to analyze its capability of handling real network traffic with acceptable model complexity. First, benefits and

	Cholesky	direct	lag zero
Number of output processes I	any	2	any
Number of input processes J	$(2M+1)I$	$(M+1)I$	I
Number of lags M	any	any	0
Number of parameters	$\frac{1}{2}J(J-1)$	$2J$	$\frac{1}{2}I(I-1)$
Parsimonious			✓

TABLE 5.2: Comparison of fitting strategies for $\mathbf{G}[m]$.

limitations are discussed and the generality of the approach is emphasized. Then, a proof of concept is presented, where models are fitted to three different types of network traffic; namely, background, online-gaming and video traffic.

5.1.4.1 Conceptual Remarks

Up to now I focused on the two most important properties of the proposed method: separability of the fitting problems and closed form analytical tractability. In the following further aspects are commented on, being of general interest in the context of source traffic modeling.

Remarks on the distribution: The distribution of the output processes $Z_i[n]$ is generated by the polynomial $\mathfrak{p}_Y(y)$, hence, it is a non-parametric distribution. The question arises of how accurately standard distributions can be represented. Of course the Gaussian distribution, as well as some other distributions (e.g., chi-squared distribution) can be perfectly resembled by a non-parametric system, since they are a polynomial transformations of Gaussian random processes. An evaluation of other standard distributions is shown in Figure 5.6 (a), where the maximum CDF distance over the polynomial order P is depicted. The $\mathcal{L}ogN(0,1)$, $\mathcal{U}(0,1)$, $\mathcal{E}xp(1)$, $\mathcal{W}bl(1,2)$ and $\mathcal{G}am(3,5)$ distributions are presented, see Appendix H for the respective definitions. As expected, a higher polynomial order leads to better fitting accuracy. Remarkably low errors are achievable for polynomials with moderate order, say P=10, which is the key feature for a parsimonious representation. The uniform and exponential distributions tend to slightly worse accuracies than other distributions, due to the point(s) of discontinuity of the respective PDFs. Further, the error-floor at roughly 10^{-5} results from the choice of percentile points $(\Omega_Z, \Omega_Y)_i$ to which the polynomials have been fitted. In the present case the first and last points correspond to the 10^{-5} and $(1-10^{-5})$ percentiles, hence, beyond those values the congruence of both CDFs is not guaranteed.

Whenever rare events are simulated (e.g., packet loss, bit errors), it is crucial that the tail of the distribution is accurately modeled, since such events are often caused by respective random samples. Hence, a maximum CDF distance of 10^{-5} may not be tolerable in the respective region. For example, by assessing connection time-outs it is important to accurately model the rare events with IATs of up to some seconds, whereas the body of the distribution with IATs in the order of milliseconds is of minor interest. In such cases, it is recommended to emphasize the region of interest by a higher density of quantile points $(\Omega_Z, \Omega_Y)_i$ and to sacrifice some accuracy in other regions. An example is given in Figure 5.6 (b), where the survival function of a Pareto distribution, $\mathcal{P}(1,1)$, is compared to the respective model fit. Due to the increased number of quantile points in the tail deployed for fitting, an acceptable fitting accuracy is obtained even for quantiles of up to $(1-10^{-8})$.

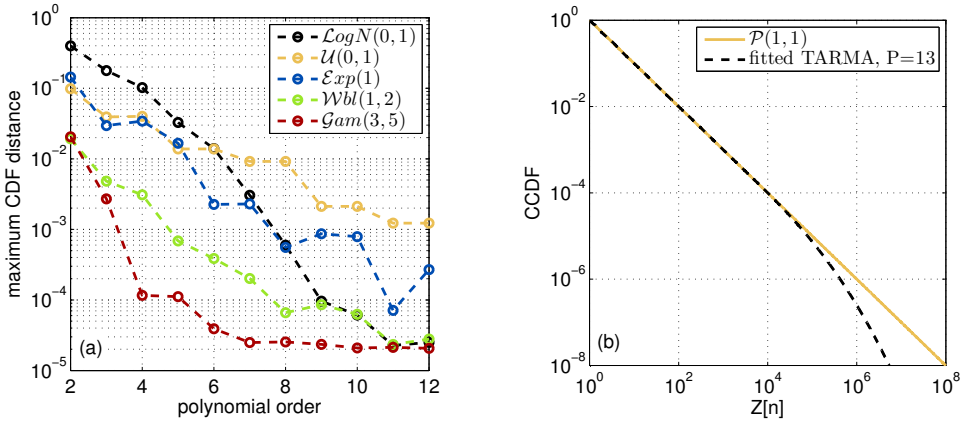


FIGURE 5.6: (a): Fitting quality of transformed Gaussian distributions over the polynomial order P of $\mathfrak{p}_Y(y)$ for various standard distributions. (b): Survival function of a $\mathcal{P}(1,1)$ distributed variable and the respective fit of a transformed Gaussian.

A further property often associated with network traffic is its occurrence with heavy-tailed distributions [168] [169]. Although this behavior applies rather to accumulated network traffic than to source traffic, it is shortly considered in the following. According to [199] at least one of the moments of a heavy-tailed distribution must not exist. However, Corollary E.3 shows that all moments of the distribution of $Z[n]$ exist and can be calculated, as long as $P<\infty$, what contradicts the heavy-tail property. Truncated heavy-tailed distributions, however, can be reproduced well by our model, as observable in Figure 5.6 (b). Even in the extreme case of a $\mathcal{P}(1,1)$ distribution (i.e., undefined mean), the model is able to sufficiently fit all quantiles up to $(1-10^{-8})$, which should be satisfactory for most simulation purposes.

Another very common property associated with network traffic is one-sided positive distributions. Gaussian random processes, however, have a domain of $\pm\infty$. Thus, the polynomial transformation should guarantee that the probability of negative values of $Z[n]$ equals zero. Due to the poor extrapolation properties of polynomials this is hardly achievable in practice. This means that in the remote case of a Gaussian sample of $Y[n]$ being close to $-\infty$ negative values of $Z[n]$ may occur. In order to absolutely prevent such cases, the values of the samples $Z[n]$ shall be limited to a minimum of zero. Theoretically, this is another non-linear transform introduced to $Y[n]$ which changes the ACFs and XCFs of $Z_i[n]$; nevertheless, due to the very low probability of occurrence of negative samples, these changes may be neglected.

Finally, network traffic may exhibit mixed continuous and discrete distributions. For example, the PS may be modeled well by a continuous distribution but exhibits a discrete number of peaks at certain common packet sizes. Such peaks are caused by routines in protocols (e.g., TCP acknowledgments). The presented modeling approach is not suited for representing single peaks, but interpolates between peaks. This behavior can be observed in Figure 5.8 (a) and Figure 5.9 (a). The advantage is that peaks resulting from a small sample size are smoothed (cf. Figure 5.9 (a)) but, on the other hand, also peaks with a concrete physical interpretation are attenuated (cf. Figure 5.8 (a)). For applications which rely on the accurate representation of a limited number of peaks, it is therefore recommended to use a different modeling approach, such as Markovian models, in this case.

Remarks on the auto-correlation function: The ACF of the process $Z[n]$ is, according to Eq. (5.4), a transformed version of the auto-correlation function of $Y[n]$. Thereby, the absolute value of the transformed ACF is always smaller or equal to the

absolute value of the original ACF, see [189, p. 133, Lemma 7.1]. Since all ACFs are defined on the interval $\mathbb{A}=]-1,1]$, both domain and codomain of the function $\mathfrak{p}_\rho(\cdot)$ equal $\mathbb{A}$, $\mathfrak{p}_\rho : \mathbb{A}\to\mathbb{A}$. However, if the image of $\mathfrak{p}_\rho(\cdot)$ in its codomain is a subset of $\mathbb{A}$, certain values of the desired ACF may not be realizable with any kind of LTI filter. Since both 1 and 0 are by definition contained by the image of $\mathfrak{p}_\rho(\cdot)$, all positive values are realizable for the ACF of $Z[n]$. Negative values on the other hand are not, whereas empirical trials suggest an increase of skewness of the CDF of $Z[n]$ to narrow the image of $\mathfrak{p}_\rho(\cdot)$ in its codomain (cf. [188]).

Long range dependencies are another typical property of network traffic [175] [171] [200]. Our modeling approach is theoretically not able to reproduce long-range dependence, since this property is equivalent to $\sum_{m=1}^{\infty}|\rho_{ZZ}[m]|=\infty$ [201], which is a contradiction to the requirement in Eq. (5.7). Nevertheless, it is possible to introduce dependencies which are arbitrary long (but less than ∞) by using ARMA models of order (1,0) or higher. This is achieved by pushing one or several roots of $\phi(B)$ close to the unit circle, which is the stability bound. An example is given in Appendix F, where dependencies up to lag $m=10^4$ are modeled very well. Furthermore, also other authors successfully approximated long-range dependencies by deploying short memory models, for example, Markovian models [180] and TES models [183].

Remarks on the cross-correlation functions: The parsimoniousness of the model $\mathbf{G}(B)$ strongly depends on the amount of output processes I and the maximum lag M up to which the XCFs shall be modeled. This requires to keep the value M small. The modeling accuracy, on the other hand, suffers from small values of M. This effect can be observed in Figure 5.7, where the XCF between PS and IAT of the *Bellcore pAug89* trace [202] is modeled. The Figure 5.7 (c) compares the real trace and its synthetic counterpart, where a maximum lag of $M=100$ is considered. In this case the modeling accuracy is very high, however a total of $J=202$ input processes $W_j[n]$ are required, with $\mathbf{G}(B)$ having more than 20 000 parameters. Figure 5.7 (b) shows the same for $M=5$. It is clearly visible that values of $\rho_{Z_i,Z_l}[m]$ are not as close to its target as in the previous case, especially for $|m|>5$. However, also values with $|m|<M$ are modeled inaccurately, which is due to the linear filters $h_i[m]$. They spread the model error of $\mathbf{G}(B)$ made for lags $|m|>M$ (concerning $X_i[n]$) over the whole range of m (concerning $Y_i[n]$). Fitting only lag zero (i.e., $M=0$) without performing the whitening operation (i.e., using $\rho_{Y_i,Y_l,\text{target}}[m]$ instead of $\rho_{X_i,X_l,\text{target}}[m]$ as input for fitting) constitutes a remedy to this problem. In this case the target XCF can perfectly be reached at lag zero, whereas errors at all other lags have to be accepted, confer Figure 5.7 (a). Notice, that the overall fitting accuracy is better than for $M=5$. Therefore, this approach is preferable compared to small values of M (but $M\neq 0$), where whitening is required. Future work has to target the problem of expressing $\mathbf{G}(B)$ in a parsimonious way for large lags M.

General Remarks: Parsimoniousness in the number of model parameters is a desired property for models, since it facilitates the reproducibility of fits and makes them less error-prone. TARMA models achieve this due to (i) the presented fitting methods allow for parsimonious fitting of each of the three statistical measures (i.e., CDF, ACF and XCF) and (ii) the separability of the fitting problem guarantees independence between the number of parameters used for each measure. For example, if a high number of parameters is required for fitting the CDF with satisfactory accuracy, this has no influence on the number of parameters required for fitting the ACF. Modeling approaches for which the fitting problem is not separable suffer from the coupling of the number of parameters (e.g., variants of Markovian models).

The computationally efficient generation of samples is guaranteed for TARMA processes.

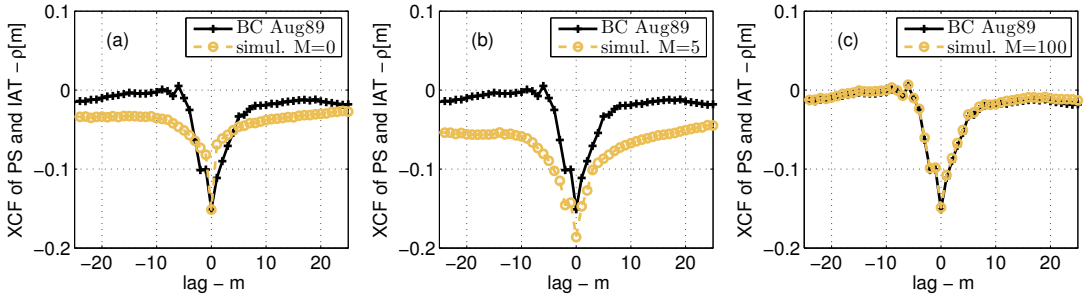


FIGURE 5.7: Fitting the XCF of PS and IAT of the *Bellcore Aug89* data set, (a): only for lag M=0, without whitening, (b): up to lag M=5, direct fitting, (c): up to lag M=100, Cholesky decomposition.

The reasons are (i) normal i.i.d. samples are efficiently generated by various known methods [165] [196], (ii) the weighting matrix requires J multiplications and additions per sample, (iii) the ARMA(P,Q) filter requires P+Q multiplications and additions and (iv) the polynomial transformation of order P requires $2P$ multiplications and P+1 additions. Summing up, each sample requires some tens of multiplications and additions, which allows for the generation of millions of samples per second on commodity hardware. Appendix G gives a respective example in *Matlab* programming language, which only requires three lines of code.

Higher order statistics appear in recent literature on traffic modeling [194], but have not been addressed in the present work. In [188] the authors discuss on possibilities how to fit respective quantities with TARMA models. Accordingly, this can be achieved by permuting the quantiles during the PDF modeling procedure (cf. Section 5.1.3.1). However, it remains unclear to which extend the uniqueness of the invertibility of the polynomials $\mathfrak{p}_{Y,i}(\cdot)$ and $\mathfrak{p}_{\rho,il}(\cdot)$ is influenced by the permutation procedure. Furthermore, the impact of various statistical measures on the modeling quality is unclear. Such measures are, for example, *bi-spectra* or *joint distribution functions* (i.e., only partially marginalized). Future work has to clarify on this issue, possibly by providing a ranking of the most important statistical measures in the context of traffic modeling.

5.1.4.2 Modeling Recorded Network Source Traffic

As a proof of concept, TARMA models for traced source traffic are presented. Thereby, three different traces are fitted, in order to demonstrate the generality of this approach. They cover (i) the popular *Bellcore Aug89* data set [171] [202], (ii) traced traffic from the online game *openarena* [203] and (iii) the online available MPEG-4 trace of the movie *Lord of the Rings I* [204]. Beside of evaluating the quality-of-fit by assessing the congruence of the three statistical measures (i.e., CDF, ACF, XCF), see Table 5.3, a benchmark is provided by feeding the traced traffic as well as respective emulated traffic to a G/G/1 queue. The complete sets of model parameters are given in Appendix G, together with an implementation example for *Matlab*.

The *Bellcore Aug89* data set [171] [202] is not typical source traffic but rather aggregated traffic, however, often used as reference for traffic modeling approaches [180] [190]. A fitted TARMA processes is presented in Figure 5.8, where PS and IAT have been modeled.The two leftmost figures present the Empirical Cumulative Distribution Functions (ECDFs) of PS and IAT, respectively. Thereby the polynomial order of the fitted transformation are both equal to P=5. It is clearly visible that the discrete steps of

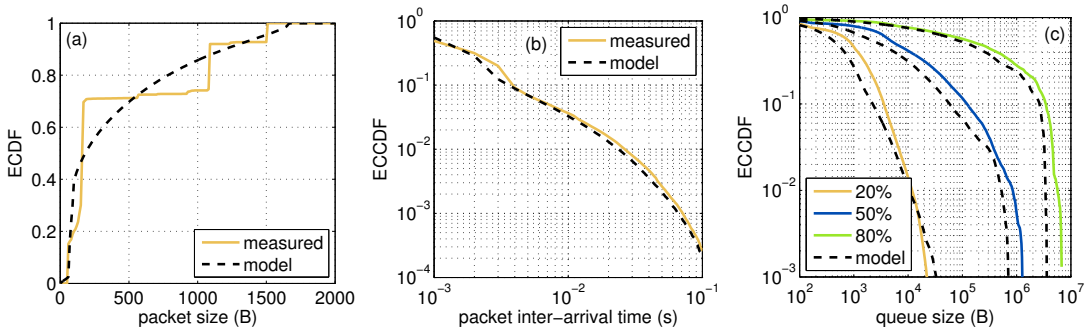


FIGURE 5.8: Fitting the *Bellcore Aug89* data set, (a): ECDFs of the packet size, note that the discrete steps are smoothed by the modeling approach, (b): survival function of the inter packet-arrival time, (c): queueing response of the traffic for various utilizations.

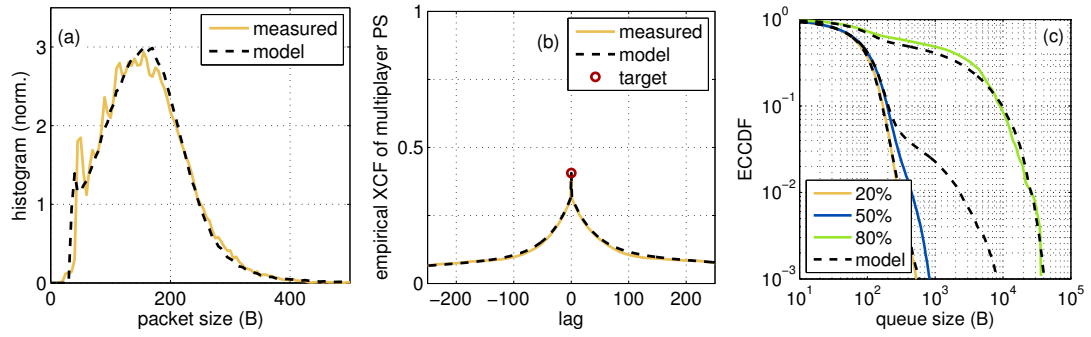


FIGURE 5.9: Fitting the *openarena* data set, (a): normalized histogram of the packet size, (b): cross-correlation function between PS processes of multiple players, (c): queueing response for different utilizations.

the PS are smoothed by the model, resulting in a relatively large maximum distance, cf. Figure 5.6 and Table 5.3. The ECDF of the IAT on the other hand is modeled well over two decades. The fits of the ACFs of the PS and IAT, and the respective ARMA(5,5) fits, are presented in Appendix F, Figure F.3 (see Appendix G for tabulated values of model paramters); they exhibit good accuracy over four decades. The XCF between PS and IAT has been modeled for several maximum lags M; shown in Section 5.1.4.1, Figure 5.7. The respective value for m=0 is negative, which is intuitively explainable by the fact that big PSs are likely to be followed by short IATs due to packet fragmentation. The rightmost plot in Figure 5.8 shows the survival function of the queuing response of the recorded and modeled traffic for different utilizations (i.e., 20%, 50% and 80%), yielding congruency for all three cases.

The *openarena* [203] data set was traced by myself, for which I sniffed IP packets in the downlink direction at a dedicated game server, which was serving two players. I observed four sessions of 10 min each, with a total of roughly 50 000 packets per player. Since the packet IAT was constant with 40 ms, only the PS is modeled, however, jointly for both players.The PDF fit is evaluated in Figure 5.9 (a), where the polynomial order equals P=5. The fitted ACF is shown in Appendix F, Figure F.4 (c), where the ARMA orders equal (5,5). The XCF is modeled only for the lag m=0, as described in Section 5.1.3, yielding a strong positive cross-correlation, see Figure 5.9 (b). Note that the XCFs is not only congruent at lag m=0, but also at all other lags up to 250. The reason is that the XCF is altered by the linear filter according to Eq. (5.11); thus, fitting the ACF

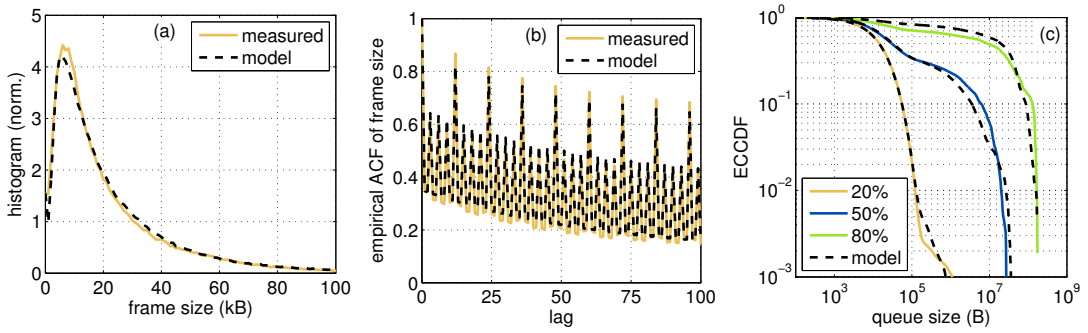


FIGURE 5.10: Fitting the *Lord of the Rings I* data set, (a): normalized histogram of the frame size (interleaved I,P and B-frames), (b): ACF of the interleaved process, (c): queueing response.

already ensures that also the modeled XCF is close to its recorded counterpart. The evaluation of the queueing performance in Figure 5.9 (c) shows, that model and real data perform very similar, except for medium utilization U=50%. This is possibly caused by the brevity of the data set, provoking inaccuracies in the estimation of the ACF at high lags and the queue length itself.

The third traffic type is MPEG-4 video traffic, whereas the online available trace of the movie *Lord of the Rings I* [204] was considered. MPEG-4 videos consist of Group of Pictures (GOPs), each of which composed of a combination of three different frame types (i.e., I, P and B-frames). In the present case the GOP exhibits a size of 12 frames according to the following structure: IBBPBBPBBPBB. For capturing this structure I modeled each frame-type as separate stream and introduced strong cross-correlation between them. The output stream was composed by interleaving the respective single streams according to the above mentioned GOP structure. Thereby, all ACFs have been equal and the cross-correlation coefficients were one ($\rho_{IP}[0]$=$\rho_{PB}[0]$=$\rho_{BI}[0]$=1), only the distributions were changed from frame-type to frame-type, cf. Appendix G, Table G.3. This approach is common in literature [154]; in the present context it can be interpreted as seasonal ARMA model [166, pp. 353ff.]. An evaluation of the interleaved output stream is shown in Figure 5.10 and Table 5.3. The leftmost plot shows the PDF which is a superposition of the PDFs of the individual frame-types. The ACF, shown in the center plot is very peaky, due to the interleaving described above. The queueing performances are compared in the right figure. All three plots exhibit a convincing quality-of-fit, cf. Table 5.3.

5.1.5 Summary and Criticism

I addressed the problem of designing a generative model for arbitrary network source traffic. Thereby I focus on multivariate stationary random processes. They shall emulate

	$F_{\mathrm{PS}}(z)$	$F_{\mathrm{IAT}}(z)$	$\rho_{\mathrm{PS,PS}}[m]$	$\rho_{\mathrm{IAT,IAT}}[m]$
Bellcore Aug89	0.2330	0.0064	0.0303	0.0239
openarena	0.0142	/	0.0626	/
Lord of the Rings I	0.0136	/	0.0491	/

TABLE 5.3: Performance evaluation, maximum distances.

certain physical quantities of the measured traffic, for example, IP packet-size or packet inter-arrival time. For each random process three statistical measures are considered, namely, the distribution, the auto-correlation function and the cross-correlation function with other processes. All of them are known for their strong influence on the network behavior.
I propose a modeling approach based on Transformed Auto-Regressive Moving-Average (TARMA) processes. This approach allows for decoupling the overall modeling problem into three independent sub-problems, one for each statistical measure. Thereby, each problem is solvable by standard techniques. The decoupling is enabled by the structure of the model, consisting of four entities: (i) a random number generator which produces normal i.i.d. random processes, (ii) a polynomial weighting matrix, introducing cross-correlations to the processes, (iii) LTI filters which introduce arbitrary auto-correlations and (iv) memoryless polynomial non-linearities which transform the Gaussian random samples to arbitrary distributions. The analytical derivations for all relevant statistical measures is feasible (cf. Appendix E), which is crucial for efficient model fitting.
Advantages of this method are its complete analytical tractability, parsimoniousness in the number of model parameters, the fitting procedure deploys only efficient standard techniques and the generation of samples exhibits low complexity.
Exemplary models for different traffic types are provided, which expose the generality of this approach. Online-gaming traffic is modeled, where packet size and packet inter-arrival times are emulated, as well as cross-correlations between multiple players. Further, a model for video traffic is shown, where the frame-size processes are emulated and combined to a single video streaming process. Beside of the applicability of the approach to network source traffic, I also indicate its usefulness for aggregated network traffic by modeling the well known *Bellcore Aug89* trace. Thereby, the packet-size and packet inter-arrival time are considered as cross-correlated random processes. In order to evaluate the proposed method by an unrelated statistical measure, the traffic traces as well as synthetic traffic were fed to a single-server queue (G/G/1 queue). The resulting queue responses show good congruency in all evaluated cases.
The focus of the present work is on accurate modeling and simple generation of synthetic traces. For ARMA models it is, to the best of my knowledge, not possible to derive any closed form solutions for theoretic queueing problems. This is in contrast to MAP models, for which respective results can be obtained [205]. Another weak point is the representation of the XCF in terms of the matrix polynomial $\mathbf{G}(B)$, which is not parsimonious at present. Future research shall address this problem. Finally, the presented method lacks any algorithm for appropriate fitting of higher order statistics, such as bi-spectra or joint distribution functions. This topic is of general interest at present, confer [188] [194]. Respective algorithms, are expected to be seamlessly integrable into the present framework.

5.2 M2M Traffic Models

In contrast to traditional Human-type Communication (HTC), which 3G wireless networks are currently designed for, M2M or Machine-type Communication (MTC) is regarded as a form of data communication that does not require human interaction [206]. MTC promises huge market growth with expected 50 billion connected devices by 2020 [207]. The support for such a massive number of MTC devices has deep implications on the end-to-end network architecture. Lowering both the power consumption and the deployment cost are among the primary requirements. This calls for a migration from high data rate networks to MTC-optimized low cost networks.

To prepare mobile networks for future requirements, standardization organizations currently investigate shortcomings of present networks by simulation of future scenarios. First studies on MTC services shine a light on such scenarios [208], the amount of deployed devices however is still far below the expected numbers, cf. [209]. Hence, a faithful definition of traffic models and reference scenarios is required, in order to validate application scenarios on current and future networks.
Conventional traffic (i.e., HTC) and MTC traffic have two major differences: (i) HTC traffic is heterogeneous whereas MTC traffic is highly homogeneous (all machines running the same application behave similar) and, further, (ii) HTC is uncoordinated on small timescales (up to minutes), while MTC may be coordinated, namely, many machines react on global events in a synchronized fashion. Thus, well known traffic models designed for HTC require adaptations for their application to MTC.
A fundamental question is whether it is feasible to model the traffic of a large amount of autonomous machines simultaneously. This approach is called **source traffic modeling**. It is in general more accurate than its counterpart, **aggregated traffic modeling** (i.e., treating the accumulated data from all MTC devices as single stream). Comparing both approaches in the context of MTC is an open issue.

5.2.1 State of the Art M2M Traffic Models

Traffic modeling means to design stochastic processes such that they match the behavior of physical quantities of measured data traffic, cf. [152]. Traffic models are classified as *source traffic* models (e.g., video, data, voice) and *aggregated traffic* models (e.g., backbone networks, Internet, high-speed links). MTC traffic fits into the second class, since the typical use case includes numerous simple machines assigned to one server or medium. This can be modeled as simple Poisson process, however, due to coordination (synchronizations) in MTC traffic, the respective arrival rate λ may be changing over time, $\lambda(t)$ (i.e., temporal modulation, [210] [211]). The more complex the single MTC devices behave (e.g., video surveillance), the more questionable becomes the approach of modeling them as aggregated traffic. The global data stream may exhibit high-order statistical properties which are difficult to capture [180]. Further, this effect is expected to be enhanced by the synchronization of sources. In such a case, traffic modeling in terms of source traffic is preferable. Source traffic models which can capture the coordinated nature of MTC traffic are available, cf. [8]. However, they are designed for a low amount of sources, thus, are too complex for MTC traffic (e.g., for N devices a $N \times N$ matrix-vector multiplication is required for each time slot, cf. Section 5.1).
Mobile networks have to adopt certain key features in order to allow MTC devices to access the air interface [212]; for example, (i) mass device transmission, (ii) uplink-dominant data traffic and (iii) small burst transmissions. Future networks shall support up to 30 000 MTC devices in one cell, which is orders of magnitude more than today's requirements [11]. Nowadays networks suffer serious Quality of Service (QoS) degradation if confronted with (i) simultaneous access attempts from many devices [209] or (ii) continuous serving of multiple devices with very low transmission duty cycle [213]. Those topics are the main focus of the research [214] [211] [215] at present.
For multiple access and capacity evaluations, aggregated traffic models such as homogeneous [214] [215] or inhomogeneous [211] Poisson processes, are a satisfactory description of reality and therefore largely deployed. Respective setups are defined in by 3$^{\text{rd}}$ Generation Partnership Project (3GPP) [209] and further discussed in Section 5.2.2. For the simulation of strongly scalable multiple access schemes in future networks (e.g., priority access, delay tolerant devices, QoS demands), mixed source traffic models have been

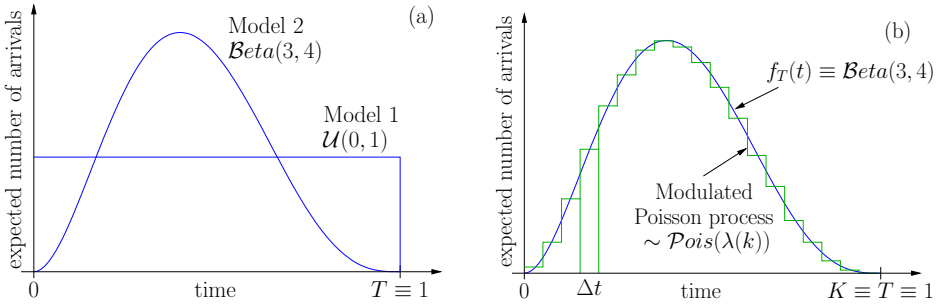


FIGURE 5.11: 3GPP MTC traffic model. (a): expected arrival rate over time. (b): interpretation as modulated Poisson process (green), sequential approach.

adopted [216] [217] [218]. In those studies synchronized MTC devices have not been considered at all [216] [217] or only for a limited number of MTC devices [218].
A divergence between traffic models deployed within different studies is observed. On the one hand higher accuracy requires source traffic models, on the other hand reduced complexity claims for aggregated traffic models. This motivates the search for refined traffic models which combine the benefits of both worlds, in order to guarantee comparability of future studies by the deployment of common models.

5.2.2 The 3GPP Model

Because of its popularity and its relation to the novel approach presented in Section 5.2.3, an overview of the 3GPP model (developed in [209]) is provided. The 3GPP model consists of two scenarios called *Model 1* and *Model 2*, the first treats uncoordinated traffic and the second synchronous traffic. Both scenarios are defined by a distribution of packet arrivals (or, equivalently, access trials) over a given time period T, cf. Table 5.4. This is shown in Figure 5.11 (a), where the PDFs of both distributions are depicted, being equivalent to the expected number of arrivals. The distributions $f(t)$ are both defined on the interval $[0, 1]$, which has to be rescaled to the time interval $[0, T]$ to yield $f_T(t)$. In order to simulate arrivals, it is sufficient to draw N samples from the given distribution and order them in time, where N is the expected number of MTC devices, cf. Table 5.4. This number may reach up to 30 000, which is the maximum amount of smart meter devices expected to be served by one cell in a densely populated urban area [209]. Definitions of the respective distributions are given in Appendix H.
In general it is undesired for simulations to generate the full traffic pattern for T beforehand. In the present case this may not be an issue, however, basic problems such as undefined run length T or large amounts of generated data, may require a sequential drawing of samples. Accordingly, time slots $k{=}1, \ldots, K$ must be defined, with a duration of Δt. This issue is discussed in [211], where it is pointed out that the 3GPP model is equivalent to a modulated Poisson process. Thereby, the modulation is achieved by

Characteristic	Model 1	Model 2
Number of devices N	1 000, 3 000, 5 000, 10 000, 30 000	
Distribution $f(t)$ over $[0, 1]$	$\mathcal{U}(0, 1)$	$\mathcal{B}eta(3, 4)$
Period T	60 s	10 s

TABLE 5.4: 3GPP MTC traffic model: Different scenarios.

the (deterministic) PDF of the arrival distribution $f_T(t) \equiv \lambda(k)$. This is depicted in Figure 5.11 (b), where a Poisson process (green – constant rate over Δt) is modulated by the beta distribution (blue). Thereby, the mean arrival rate $\lambda(k)$ of a Poisson process is adjusted to a beta distribution in each time slot k (modulation). Here K corresponds to the simulation duration T. For infinitesimally short time slots, $\Delta t \to 0$, both curves coincide. Consequently, sequential sampling is performed by the generation of a Poisson distributed number of arrivals in each time slot with mean arrival rate $\lambda(k)$. In order to obtain an expected outcome of N samples within the period $K \equiv T$ (i.e., one sample per machine), the arrival rate has to be normalized according to $\lambda(k) = f_T(k \cdot \Delta t) \cdot \frac{\Delta t}{T} \cdot N$. The algorithms for the two different sampling strategies are outlined in Figure 5.13 (a–b).
The 3GPP model reaches its limits for further requirements such as:

- the amount of machines becomes lower, so that a data source has to be associated with a fixed location,
- multiple packets (bursts) shall come from the same machine,
- the synchronous traffic (*Model 2*) influences the regular traffic (*Model 1*) and
- the network has an influence on the traffic patterns (e.g., the devices are forced to suppress delay tolerant traffic).

5.2.3 The CMMPP Source Modeling Approach

In order to circumvent the limitations of the 3GPP model, the source modeling approach has to be adopted. This means that each MTC device is represented by a separate entity in the model. Thereby, a trade-off between mutual couplings among data sources (synchronization) and tolerable complexity for large amounts of devices has to be found. Generic traffic models introduce couplings by bidirectional links between devices (cf. Section 5.2.1) which would be too complex for the present purpose. Instead, one background process acting as *master* is proposed, which modulates all MTC device entities. In the following Markov Modulated Poisson Processs (MMPPs) are presented as models for single MTC devices. Due to their simplicity the operation of large amounts of device models in parallel is computationally feasible. Further, the coupling to a master process with low complexity is possible.

5.2.3.1 MMPP Basics

Markov models and Markov modulated Poisson processes are common in traffic modeling and queueing theory, since they allow for analytically tractable results for a broad spectrum of use cases, cf. [24] [210] and Section 5.1.1. MMPP models consist of a Poisson process modulated by the rate $\lambda_i[k]$, which is determined by the state of a Markov chain $s_n[k] = 1, \ldots, i, \ldots, I$. This principle is depicted in Figure 5.12, where $p_{i,j}$ are the transition probabilities between the states of the chain. In the present source modeling approach each MTC device n out of N is represented by a Markov chain and a corresponding Poisson process. The state transition probabilities are condensed into the state transition matrix $\mathbf{P}$ and the state probabilities π_i into the state probability vector $\boldsymbol{\pi}$ according to

$$\mathbf{P} = \begin{pmatrix} p_{1,1} & p_{1,2} & \cdots \\ p_{2,1} & p_{2,2} & \\ \vdots & & \ddots \end{pmatrix} \qquad \boldsymbol{\pi} = \begin{pmatrix} \pi_1 \\ \pi_2 \\ \vdots \end{pmatrix}. \tag{5.21}$$

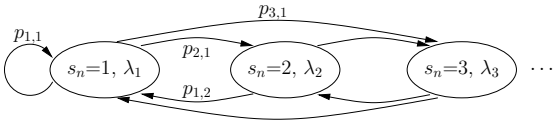


FIGURE 5.12: The MMPP model: each MTC device n is represented by a Markov chain with states s_n, which inherit the parameter λ_i. This is the mean arrival rate, modulating the respective Poisson process.

In the stationary case both are related by the balance equation $\boldsymbol{\pi}=\boldsymbol{\pi}\mathbf{P}$, which yields $\boldsymbol{\pi}$ an eigenvector of $\mathbf{P}$ to the eigenvalue of 1. Further, the global rate of the MMPP calculates to $\lambda_g=\sum_{i=1}^{I}\lambda_i\pi_i$, where I is the total number of states. A basic example for an MTC device modeled by a MMPP would be a two state MMPP with the first state representing *regular* operation, the second *alarm*. This is analogous to the 3GPP model presented in Section 5.2.2.

5.2.3.2 Coupling Multiple MMPP

State transition matrix $\mathbf{P}$ remains to be determined such that each device model resides a dedicated amount of time in the regular and alarm states. From the perspective of a single device, this may be an easy task; however, from a global perspective, the devices in the 3GPP model perform the transition from the regular to the alarm state in a strongly correlated manner, both in time and space. To correlate multiple MMPP models, they must be coupled.
Coupled Markov chains, as introduced in the context of pattern recognition [219] [220], are multiple chains which mutually influence their transition probability matrices $\mathbf{P}_n[k]$; whereby k denotes sampling instances in time. This is achieved by the multiplication of the respective matrices with weighting factors $\gamma_{n=i|m=j}[k|k-1]$, which depend on the past states $s_m[k-1]$ of other chains m (i.e., background processes).
For the present purpose only unidirectional influences from a background process (*master*) $\Xi(t)$ to the MMPP models of each MTC device are considered. I name this approach Coupled Markov Modulated Poisson Processes (CMMPP). To avoid the separate tuning of each of the parameters $\gamma_{n=i|\Xi(t)=j}[k|k-1]$ for each machine, they are set into the following framework:

Definition 5.1 (*Coupling of MMPPs*)

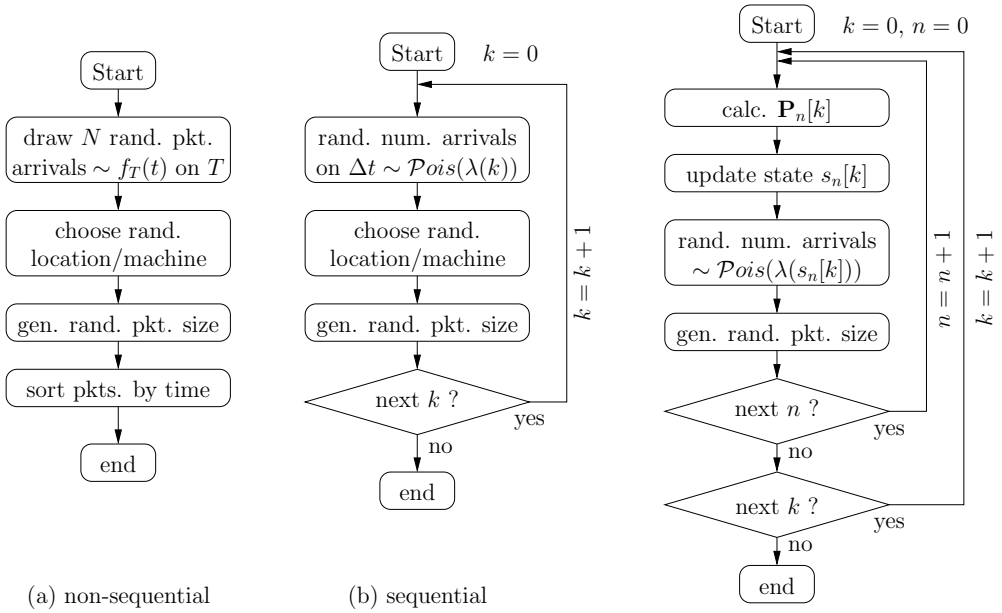
Let there be two transition matrices $\mathbf{P}_C$ and $\mathbf{P}_U$ globally valid for all N MMPP models and a background process $\Xi(t)$, producing samples $\xi[k]$ from the interval $[0,1]$, were $t=k\cdot\Delta t$. Further, a parameter $\zeta_n \in [0,1]$, constant over time, is associated to each MTC device n yielding

$$\xi_n[k] = \zeta_n \cdot \xi[k].$$

Then the state transition matrix $\mathbf{P}_n[k]$ shall be calculated for machine n at time slot k according to

$$\mathbf{P}_n[k] = \xi_n[k] \cdot \mathbf{P}_C + (1-\xi_n[k]) \cdot \mathbf{P}_U.$$

This form is a convex combination of both transition matrices, yielding itself a valid transition matrix. The advantage is that instead of tuning an enormous amount of parameters $\gamma_{n=i|\Xi(t)=j}[k|k-1]$, only a single global parameter $\xi[k]$ has to be generated and can be applied for all device models n. The matrices $\mathbf{P}_C$ and $\mathbf{P}_U$ can be interpreted



(a) non-sequential 3GPP model

(b) sequential 3GPP model

(c) CMMPP model

FIGURE 5.13: Flow diagrams for traffic generation by the three different models presented in this work, ordered by computational complexity.

as transition matrices for the case of perfectly coordinated devices and uncoordinated devices, respectively. The parameter ζ_n can be interpreted as closeness (distance) to the epicenter. The closer $\xi_n[k]$ to zero, the more uncoordinated the respective machine behaves; the closer $\xi_n[k]$ approaches one, the stronger is the coordination. Further, $\Xi(t)$ may have an infinite amount of states, yielding $\xi[k]$ a continuous process. The global arrival rate λ_g equals

$$\lambda_g = \sum_{k=0}^{K} \sum_{n=1}^{N} \sum_{i=1}^{I} \lambda_i \pi_{n,i}[k], \tag{5.22}$$

however, the calculation of this expression is rather involved, since $\boldsymbol{\pi}_n[k]$ changes for each time instant k and device n. Unlike the transition probability matrix $\mathbf{P}_n[k]$, the state probability vector is not a convex combination of $\boldsymbol{\pi}_C$ and $\boldsymbol{\pi}_U$, but a rational function in $\xi_n[k]$ with degree $I-1$.

The generation of arrivals according to the CMMPP model is outlined in Figure 5.13 (c). Two iteration loops are required, both for the devices n and time instances k, respectively. In each iteration the transition matrix $\mathbf{P}_n[k]$ is calculated anew according to Definition 5.1. This may appear expensive, however, can be computed efficiently, since it is a convex combination of a low amount of matrix entries. Then the random state update from $s_n[k-1]$ to $s_n[k]$ is performed. Afterwards, a number of arrivals and packet sizes are generated appropriate to the actual state $s_n[k]$. A comparison of the complexity of the three approaches is provided in Table 5.5 and illustrated in Figure 5.15.

5.2.3.3 Deployment Example

To emphasize the convergence between the 3GPP model and the CMMPP model, I consider the following deployment example: Assume a two-state Markov model for each MTC device, with State 1 representing *regular* operation and State 2 *alarm* operation.

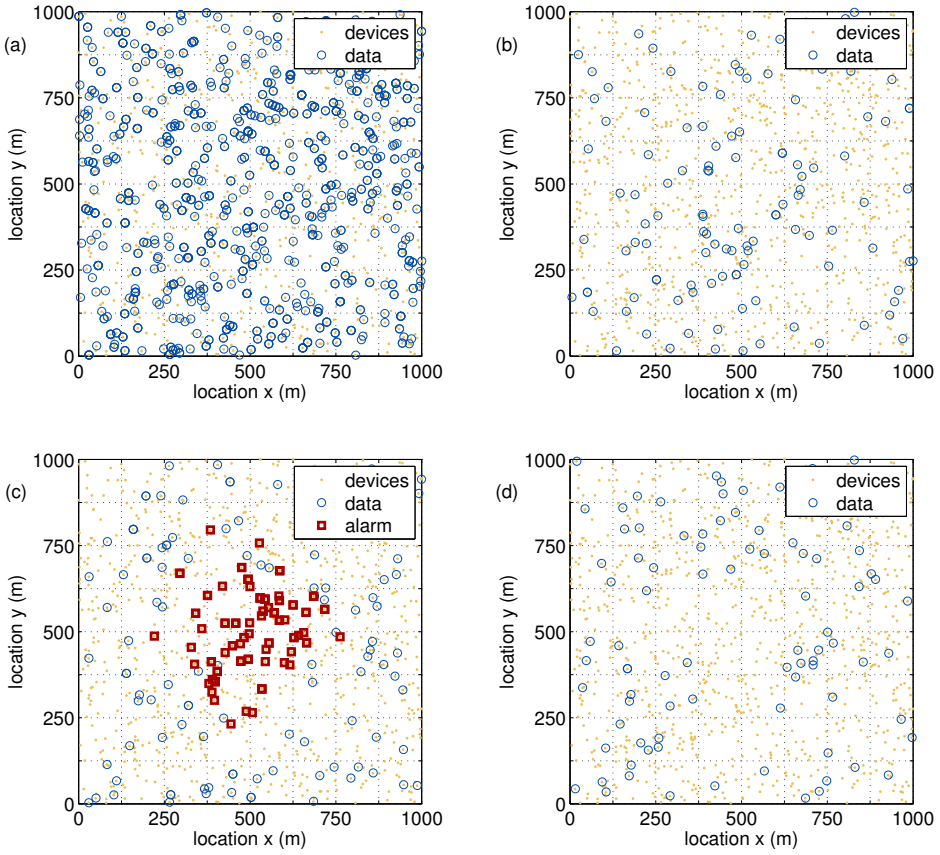


FIGURE 5.14: Deployment of the 3GPP model with basic extensions: 1 000 MTC devices, 60 s runtime, regular operation with 17 pkt/s/km^2, four different states: *startup*, *regular*, *alarm*, *silent*. Legend: (i) device: location of a device, (ii) data: the device transmits data, (iii) alarm: the device transmits an alarm. (a): startup phase, second 0–7. (b): regular operation phase, second 16–23. (c): alarm phase, second 26–33. (d): silent phase, second 40–47, note the low activity in the center, since all devices which issued an alarm are silent at this phase.

Thereby, λ_1=0.0005 pkt/s/device and $\lambda_2=\frac{1}{\Delta t}$ pkt/s/device. The global transition matrices are defined to

$$\mathbf{P}_U = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} \qquad \mathbf{P}_C = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \tag{5.23}$$

where the functionality for the uncoordinated state is to never trigger an alarm and for the coordinated state to trigger one alarm in one time slot and then to return to regular operation. The process $\xi[k]$ was fixed to $\xi[k]=f_T(k\cdot\Delta t)\cdot\frac{\Delta t}{T}$, namely, the PDF of the beta distribution of *Model 2* of the 3GPP model, scaled by the number of time slots. This is convenient since for a high amount of short intervals ($\Delta t \ll T$) the function $\xi[k]$ becomes small (close to zero). Consequently, the state probability vector $\boldsymbol{\pi}[k]$ can be approximated as linearly dependent on $\xi[k]$, instead of considering a rational polynomial function, cf. Section 5.2.3.2. The probability of residing in the alarm state estimates to $\pi_2[k]\approx\kappa\ \xi[k]$. By scaling the value λ_2 (or $\xi[k]$ itself) according to κ, it is easily achieved to trigger approximately one alarm per machine during the whole simulation/emulation run. Finally, the closeness function ζ_n was fixed to the values of Gaussian PDFs, scaled to one at the epicenter. The results of a respective traffic emulation closely resembles the 3GPP model, however, with superimposed uncoordinated traffic (*Model 1*) and synchronous traffic (*Model 2*). The model comparison provided in Section 5.2.4 is based on

this example.
In a further example, the CMMPP model is augmented by two states, namely, State 3, representing the *startup* phase, and State 4, representing an extended *silent* phase of the MTC device after having issued an alarm. In this simulation setup the strengths of the CMMPP modeling approach become explicit. Namely, multiple devices are able to pass state sequences in a spatially/temporally coordinated fashion. For illustration, snapshots at different time instances of a simulation run are depicted in Figure 5.14. Four different phases corresponding to the four state of the CMMPP are clearly distinguishable: (i) during the *startup* phase each device tries to transmit information, (ii) in the *regular* phase sparse uncoordinated traffic is generated, (iii) the *alarm* phase triggers affected devices to change their state, whereas the others stay in regular operation, and (iv) during the *silence* phase all devices which issued an alarm do not transmit. This last phase can be distinguished from the *regular* phase by the low activity in the central region of the bottom right figure, compared to the activity in the respective area of the top right figure. Such correlations between spatial and temporal activities are far beyond the capabilities of the 3GPP model.

5.2.4 Model Comparison

Most advantages and drawbacks for both the 3GPP (non-sequential and sequential) and the coupled MMPP models have already been discussed in Section 5.2.2 and Section 5.2.3, respectively. For completeness a summary and comparison of the models is given in Table 5.5.
For comparing the computational complexity of both models, the basic example from Section 5.2.3.3 has been considered, which is similar to the plain 3GPP model. The simulated time was 60 s with a resolution of 10 ms. The three models have been implemented in *Matlab* (available [121]) and the respective simulation durations on a commodity desktop workstation were recorded. The resulting absolute numbers for the emulation of 30 000 devices are: 0.02 s, 1.1 s and 36 s for the *3GPP*, *3GPP seq.* and *CMMPP* model, respectively. The result for CMMPP positively answers the general question of the feasibility of source modeling approaches for large numbers of sources. A comparable simulation with conventional source traffic models (cf. Section 5.2.1) would be unfeasible, since it requires roughly 20 h for 30 000 devices. In this context the term

Model Type	3GPP aggreg.	3GPP seq. aggreg.	CMMPP source	Generic source
Complexity	low, $\mathcal{O}(1)$	medium, $\mathcal{O}(1)$	high, $\mathcal{O}(N)$	unfeas., $\mathcal{O}(N^2)$
Temporal coord.	✓	✓	✓	✓
Spatial coordination	✓	✓	✓	✓
Temp./Spatial coord.			✓	✓
QoS possible	✓	✓	✓	✓
Determ. sample path	✓	✓	✓	✓
Random sample path		✓	✓	✓
Random run time		✓	✓	✓
Fixed device location			✓	✓
Coupling traffic states			✓	✓
Reciprocal dev. coupl.				✓

TABLE 5.5: Comparison of the four models.

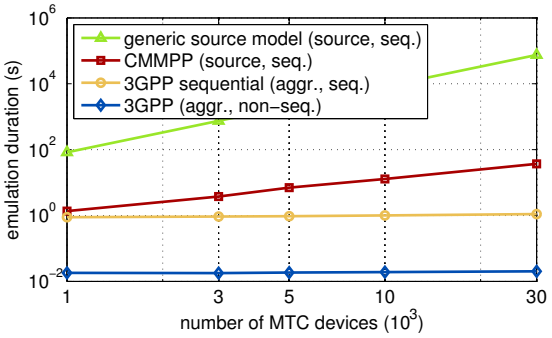


FIGURE 5.15: Comparison of the simulation/emulation duration (*Matlab*) of the two 3GPP models, the CMMPP model and a conventional source traffic model (estimate). The first two are aggregated models, the second two source traffic models. The simulated time is 60 s with a resolution of 10 ms.

unfeasible denotes the fact that the simulation duration is much longer than the simulated period. A visual comparison is provided in Figure 5.15. In general the sequential approaches perform slower than the non-sequential. Both 3GPP models show a negligible raise in complexity with increasing number of MTC devices, which is expected for accumulated traffic models. The CMMPP approach exhibits a linear growing complexity with the number of devices, since each device is internally represented by a separate MMPP model. Conventional source traffic models experience a quadratic growth in N (cf. Section 5.2.1).

5.2.5 Summary and Criticism

Existing traffic models for MTC or M2M communications are mostly *aggregated* traffic models, defining MTC traffic as one stream from multiple devices (e.g., see 3GPP [209]). For a more accurate description of data traffic, *source* models are required, which model each MTC device on its own. Yet, M2M traffic bears two fundamental problems to source modeling: (i) the massive amount of devices to be modeled in parallel and (ii) the strong spatial and temporal correlation between the devices.
In this section I propose CMMPP models for MTC traffic to overcome those problems. This solution allows for involved correlation structures, exceeding the capabilities of the 3GPP modeling approach. It is tailored to MTC traffic in the sense that (i) it assumes a huge amount of individual traffic sources, all being equal and behaving simple on their own (i.e., few Markov states, cf. Section 5.2.3.3) and (ii) it incorporates one (or a few) background processes which introduce involved temporal and spatial correlations among single devices.
The generation of multiple MMPPs exhibits low computational cost, such that their massive parallel deployment is feasible. The coupling to a background process is achieved by a convex combination of multiple state transition matrices. A complexity evaluation of the proposed model emphasized its usefulness as it demonstrates the parallel deployment of 30 000 machines with reasonable effort. Thus, source traffic modeling is feasible for MTC traffic. In elaborate scenarios and for a low or medium number of devices CMMPP are preferable over aggregated traffic models for their higher achievable accuracy.
The strength of the proposed model is the possibility to couple (correlate) separate MTC devices. This fact requires all machines to have a common notion of time, or, in algorithmic terms, to perform frequent periodic updates of the Markov models of each machine. Given the huge amount of simulated M2M devices (being idle most of the

time), it would be preferable to follow an approach where the Markov models have to be updated only at the time of the transmission of a packet. This can be achieved by *semi-Markov models*, as presented in [221], which however prevents any coupling between devices. Consequently, the extension of the proposed method to semi-Markov models is highly desirable and subject to future work.

5.3 Background Traffic

User traffic unquestionably affects the network performance. The strong growth of users in mobile cellular networks raises the need to understand their data traffic statistics on cell level, in order to plan and run networks efficiently. Performance evaluations based on simulations should rely on realistic traffic patterns. Those can either be directly deployed for the performance assessment or indirectly, by simulating a cell with realistic background users. Nevertheless, for simplicity, lack of standardization and historical reasons, most simulations are performed assuming a constant number of users with full buffers [222] [223] [224]. The 3GPP makes up for this deficiency by specifying statistical models for different kinds of data-traffic in combination with a model for the traffic-mix [225]. However, an implementation of those may be very complex.
This section provides a behavioral analysis of mobile cellular users. This is achieved by evaluating user payload captured in a live HSPA network on the Iub interface (i.e., Probe 3 in Figure 3.2). The measurements cover the overall downlink data of approx. 400 cells over three days. The inquiry took place in November 2010 in the city of Vienna, as explained in Section 5.3.2. Various statistics are identified to be of interest for numerical simulations, see Section 5.3.3. Such include the overall cell throughput over daytime, the number of users per cell and the throughput per user and user-session duration. Section 5.3.3.1, Section 5.3.3.2 and Section 5.3.3.3 provide respective evaluations. Furthermore, empirical models are presented for the simple generation of realistic traffic patterns. Those can generate active user traffic as well as cell background traffic.

5.3.1 Related Work

Analysis of network traffic was developed in the context of telephone networks and computer science. Concerning cellular networks, this topic gained attention in course of admission control for 2G and 3G technologies. Since the traffic in cellular networks was mainly consisting of voice calls at that time, many concepts have been borrowed from wired telephony. In [226], for example, the authors model each cell as independent $M/G/\infty$ queue, resulting in Poisson random processes. Voice calls differ from data traffic in many aspects, for example throughput and burstiness of the traffic. For cellular data traffic an early measurement study has been carried out on a wired network [227], where the authors use traced Internet traffic on which they impose restrictions (e.g., on throughput) tailored to mobile networks. Furthermore, public wireless LAN traffic, comparable to cellular data traffic, has first been studied in [228]. In that work, user fluctuation, traffic share and typical session duration have been reported. A more recent inquiry [229] analyzes the cellular user behavior on a large scale; the main interest of it, however, is spectrum occupancy of the UMTS bands for feasibility of cognitive radio. Another detailed large-scale measurement evaluation of traffic-dynamics is given in [230], where mobility, daily and weekly periodicity, user throughput distributions and traffic share are examined. The authors of [231] focus on web traffic and present a survey on traffic share, in combination with a respective classification algorithm. Simulations of

web traffic are reported in [232], where the capacity of HSPA cells is evaluated in terms of number of users.

5.3.2 Measured Data Set

Large-scale measurements in a live 3rd Generation (3G) network have been performed by monitoring user-generated HSDPA traffic at Iub-links, located between Base Stations (NodeBs) and the respective Radio Network Controllers (RNCs); corresponding to Probe 3 in Figure 3.2. For this purpose a passive monitoring tool, called *METAWIN*, was deployed (cf. [124] [96] [126]). The measurement software anonymizes the captured data in order to satisfy privacy requirements as outlined in Section 3.2.2. This is necessary since the data stems from real users.
For the present analysis only downlink High Speed Downlink Packet Access (HSDPA) user data was extracted from the Radio Link Control (RLC) protocol layer. It was recorded over a period of 90 hours ($\sim$3.5 days). Thereby, 112 randomly chosen NodeBs connected to one RNC have been monitored, covering a metropolitan area within the city of Vienna, Austria. Those NodeBs serve $\sim$400 cells (sector-cells), which provide a downlink data rate of up to 14.4 Mbit/s. Since the recording covers only HSDPA traffic, it is impossible to analyze the amount of voice users in the cell. Hence, although having an influence on the admission of data users, they are not part of this study.
The large amount of data coming from the Iub interface is preprocessed and stored in an intermediate database. The throughput information is necessarily aggregated over one second per user, cell and NodeB. Besides performance reasons, the aggregation time of one second was chosen because it is short enough to temporally resolve human behavior (e.g., in terms of web-browsing) but also long enough to average over synchronization and bundling effects introduced by network components. The measured throughput volume is corresponding to RLC payload, measured in bits; hence, rather comparable to data transmitted at the air-interface than to effective higher protocol user-payload.
Users are distinguished by means of the UTRAN Radio Network Temporary Identifier (U-RNTI), which has the advantage of being explicitly transmitted at the Iub interface for all packets. However, the drawback of this identifier is its limited temporal validity, for example, one user could appear after a short idle time ($<$1 min) with a different U-RNTI. Therefore, the user behavior is evaluated in terms of sessions. A user-session consists of all the consecutive seconds in which the same U-RNTI transmits at least one bit. After a pause of at least one full second, a session is considered to be closed and the same physical user may start a new session (which would correspond to a new user). In the following the terms *user*, *session* and *U-RNTI* are used interchangeably. The presented data set comprises more than twelve million sessions.

5.3.3 Evaluation and Results

The choice of parameters extracted from the measurement data influences (i) the resulting models as well as (ii) the type of simulations those models are applicable for. Therefore, the requirements on the models are analyzed before the parameters under investigation are defined. Three simulation parameters are found to be considerable for model building; I arranged them into the set

$$\Sigma = \{sim\_duration,\ num\_users,\ user\_traffic\}, \tag{5.24}$$

where *sim_duration* specifies the simulated time in terms of *snapshot* ($<$ 3 s) or *long-run* ($>$ 3 s) simulations, *num_users* defines statistics for the number of users per cell,

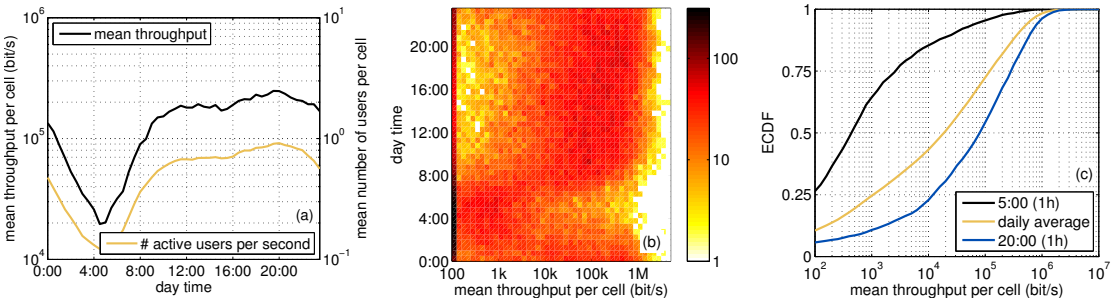


FIGURE 5.16: (a): Mean throughput and mean number of active users per second and cell over daytime. (b): Histogram of mean throughput per cell for different daytime (colors in logarithmic density). (c): ECDFs of mean throughput per cell, various daytimes.

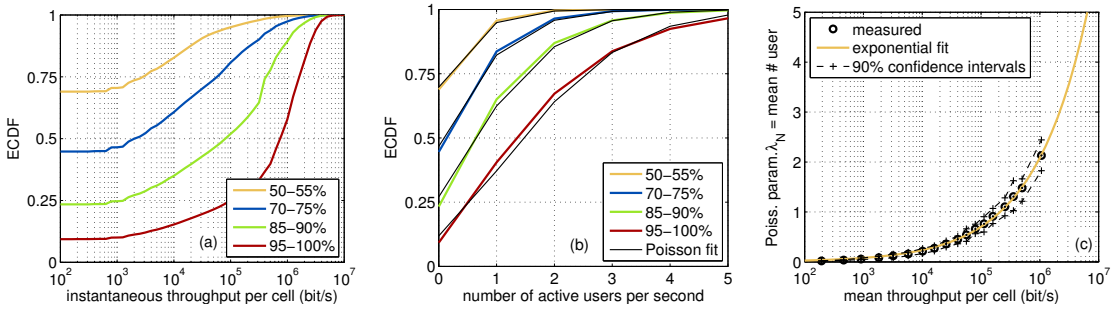


FIGURE 5.17: (a): ECDFs of instantaneous (per second) throughput of all cells for different load. (b): ECDFs of instantaneous (per second) number of active users per cell for different load. Additionally, fit of a Poisson-CDF for each ECDF. (c): Poisson-parameter (λ_N) matching the ECDFs of the central figure best, over mean throughput (30 min).

for example, *constant* or *Poisson* distributed and, *user_traffic* provides the type of user traffic, for instance, *greedy* (full buffer) or *background* traffic. As mentioned at the beginning of this section, the most common simulation setting is {snap, const, greedy}. In the following more realistic models are determined, both for the number of users and the respective traffic type.

The evaluation approach is to analyze the data at three different time scales, namely, 1 s, 30 min and 1 day. Cell statistics are evaluated over one day, with aggregation of 30 min, see Section 5.3.3.1. Thereby, I focus on the average throughput over 30 min (denoted *mean throughput* or *relative load*). Further, 30 min time series of cells are clustered according to their relative load and the respective behavior is analyzed with 1 s resolution, see Section 5.3.3.2. Thereby, the parameters of interest were: (i) the number of users and (ii) the throughput (denoted *instantaneous throughput*). Finally, user traffic statistics are extracted independent of the relative load with a resolution of 1 s, see Section 5.3.3.3.

5.3.3.1 Network Level Results

The following results show relations between the load of single cells and the load of the whole network. First, the peak hour of the network is analyzed. Therefore, the average of the accumulated data within intervals of 30 min is calculated. The outcome is depicted in Figure 5.16 (a) for throughput and number of users over daytime, respectively.

This corresponds to an average cell on an average day. It is clearly visible, that the mean number of active users within one second and the mean throughput are strongly correlated. Both have their minimum around 5:00 and their maximum around 20:00. The difference is in the order of one magnitude, respectively. However, in this case the mean as measure has limited significance, since the variance (over cells) of the cell-load is strongly asymmetric. This is illustrated by Figure 5.16 (b), showing histograms of cell-load (x-axis) over the daytime (y-axis). The frequency of occurrence is expressed by the color on a common logarithmic scale (e.g., orange $\sim 10^1$ cells, dark red $\sim 10^2$ cells). Notice, that a significant amount of cells has a mean throughput below the global mean (e.g., corresponding to 250 kbit/s at 20:00), with the extreme case of hundreds of idle cells at 5:00 (the leftmost column of bins, labeled "100 bit/s", also includes the idle or 0 bit/s case). The ECDFs for three different cases are shown in Figure 5.16 (c): (i) the hour with the lowest mean throughput (4:30-5:30), (ii) the daily average and (iii) the peak hour (19:30-20:30). For all three cases a huge variation in mean throughput is observed. The distance between the 10th and the 90th percentile is almost three orders of magnitude for each case. These curves show that the common assumption of equal average throughput and user density in all cells of the network (often made for simulations) does not reflect reality.

5.3.3.2 Cell Level Results

For the analysis of instantaneous user behavior the separation of the data set of single cells into temporal intervals of 30 min is preserved. The ECDF of the mean throughput in Figure 5.16 (c), labeled *daily average*, is the accumulation of each of such intervals of all cells. Therefore, I define the *relative load* of a cell in the given time to the value of this ECDF resulting from the respective mean throughput (cf. Section 5.3.3). Statistics for specific values of relative cell-load are extracted from this definition, by considering only those 30 min intervals independent of the cell or daytime.
Corresponding results are shown in Figure 5.17. In the figure to the left, ECDFs of the instantaneous throughput are depicted for different relative cell-load factors. The amount of idle seconds, even at full cell-load (95–100%), is remarkable. Figure 5.17 (b) shows ECDFs of the number of active users per cell and second, N_u. Also in this case different load factors are examined. Additionally, Poisson distributions are fit to the ECDFs with different Poisson parameters λ_N for each load factor, see Appendix H for details. The estimation of λ_N, as well as all other parameters mentioned below, was performed deploying the method of moments. For assessing the quality of the fit, Cramer's distance [131] is deployed

$$D_C(X,Y) = \int_{-\infty}^{\infty} \Big(F_X(\zeta) - F_Y(\zeta)\Big)^2 d\zeta, \tag{5.25}$$

where X and Y are two random variables and $F_X(\zeta)$ and $F_Y(\zeta)$ the respective CDFs. For the 95–100% loaded cells it calculates to $D_C{=}3{\cdot}10^{-3}$ and is decreasing for lower load (e.g., $D_C{=}1.4{\cdot}10^{-4}$ at 50–55%). This approves the assumption of modeling each cell as $M/G/\infty$ queue, as proposed in [226] and references therein.
Figure 5.17 (c) shows points for different such Poisson fits, in terms of Poisson parameter λ_N over mean throughput per cell. Each point presents a mutually exclusive 5% cell-load interval. The plot exhibits a strong correlation between its variables, with a surprisingly high concentration of the measurement values. The estimates for λ_N follow an exponential function; hence, reveal strongly non-linear dependence on the average

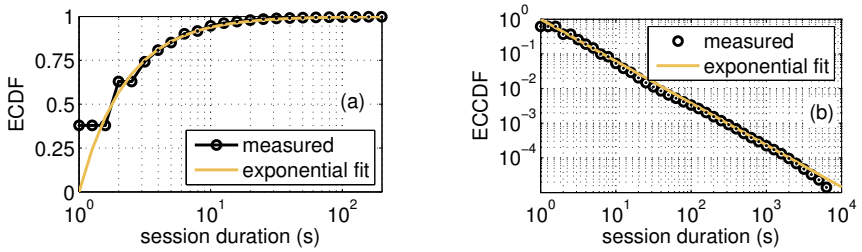


FIGURE 5.18: (a): ECDF of user-session duration on logarithmic scale with fit of an exponential distribution. Measurement resolution of 1 s. (b): ECCDF of session duration on double-logarithmic scale with exponential fit. Asymptotic slope (Pareto index) α_L=1.21.

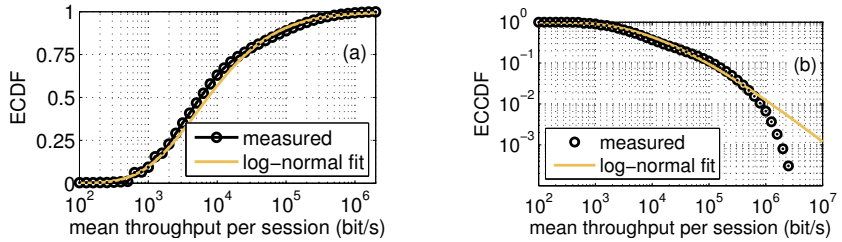


FIGURE 5.19: (a): ECDF of mean throughput per user-session (total throughput divided by session duration) on logarithmic scale with log-normal fit. (b): ECCDF of mean throughput per session on double-logarithmic scale. Asymptotic slope (Pareto index) α_R=0.95.

throughput. Summarizing those characteristics of the number of active users yields

$$N_u \sim \mathcal{P}ois(\lambda_N); \qquad \lambda_N = 0.0031 \cdot e^{1.085 \cdot \log_{10}(\bar{R}_c)}, \tag{5.26}$$

whereas N_u is the number of active users, λ_N denotes the Poisson parameter and $\bar{R}_c$ denotes the mean desired throughput per cell in bit/s. This finding is valuable for simulations, in order to obtain a definite and realistic amount of users to be placed in one cell for a desired average throughput. The possibility of extrapolating the measurement data is especially attractive in case of simulating future networks. Nevertheless, the Poisson distributed number of users, obtained by Eq. (5.26), yields the desired average throughput only if the individual user behavior (see Section 5.3.3.3) is designed accordingly.

5.3.3.3 User-Session Level Results

The user behavior on session level is explained in the following (for definition of *session* see Section 5.3.2). All respective statistics are extracted without distinction of different cell loads the users faced during their session, but averaged over all scenarios. For characterizing the user behavior in a simple and reconstructible way, two features are chosen to represent a user-session; those are (i) the session duration L_s and (ii) the mean throughput of the session $\bar{R}_s$. Any analysis of further characteristics of the data (e.g., burstiness) is omitted, because of the limited temporal resolution (1 s). An evaluation of the session duration is presented in Figure 5.18. The left plot shows the ECDF of the session duration. Because of the temporal resolution of 1 s the ECDF exhibits steps at the left end. It resembles an exponential CDF, cf. Appendix H, in logarithmic scale. A

random variable

$$L_s = 10^{\Delta}; \qquad \Delta \sim \mathcal{E}xp(0.3591) \tag{5.27}$$

fits the empirical data closest, with an error of D_C=$43.7 \cdot 10^{-3}$, cf. Eq. (5.25). Thereby, L_s denotes the session duration and Δ denotes an exponential distributed random process. The resulting distribution of L_s exhibits heavy-tail characteristics [233], namely,

$$P\{L_s > l\} = 1 - F_L(l) \sim l^{-\alpha_L}, \tag{5.28}$$

where $P\{\cdot\}$ denotes the probability, $F_L(l)$ the CDF of L_s and α_L the Pareto index, cf. Appendix H. The Pareto index is determined by graphical inspection of the survival function or CCDF of L_s in double-logarithmic scale, see Figure 5.18 (b). The asymptotic slope of the survival function appears to equal α_L=1.21. Consequently, the mean of L_s exists (α_L>1) and calculates to $\bar{L_s}$=5.71 s, whereas the variance is not existing (α_L<2). The low value of the Pareto index is caused by rare sessions which exhibit very long duration. This implies that simulations have to span long periods in order to capture a statistically significant amount of such heavy sessions.
Combining this result with Eq. (5.26) by using Little's law (see Theorem 2.3, [24]) allows to determine the arrival rate A_s of users (sessions), yielding

$$A_s \sim \mathcal{P}ois(\lambda_A); \qquad \lambda_A = \frac{\lambda_N}{\bar{L_s}} = \frac{\lambda_N}{5.71\,\mathrm{s}}, \tag{5.29}$$

with the Poisson parameter for arrivals λ_A, the Poisson parameter for the number of active users λ_N, and the mean session duration $\bar{L_s}$, see Appendix H for details on the distributions.
The second feature of interest, the mean throughput of a session, is evaluated in Figure 5.19. Again the left plot shows its ECDF. The steps at the very left of the function arise from certain packet sizes which occur frequently, for example, due to protocol definitions (e.g., TCP-acknowledgment packets). The measured data conforms to a log-normal distribution on logarithmic scale, visualized by the curve fit (solid line). The resulting distribution can be reproduced by

$$\bar{R_s} = 10^{\Phi}; \qquad \Phi \sim \mathcal{L}ogN(1.3525, 0.1954), \tag{5.30}$$

with the mean throughput per user-session $\bar{R_s}$ in bit/s and the log-normal distributed random process Φ. Cramer's distance equals D_C=188 for this fit. According to Eq. (5.28) also this distribution is heavy-tailed, again visualized by plotting the empirical survival function (ECCDF) on a double-logarithmic scale, see Figure 5.19 (b). Note the steep drop of the measurement values at throughput above 1 Mbit/s. This is due to a truncation effect on the Pareto-like tail of the distribution of $\bar{R_s}$, see [233], which is caused by the maximum throughput of the cells (up to 14.4 Mbit/s). The Pareto index of the distribution is estimated to α_R=0.95, which implies that the mean of the random process $\bar{R_s}$ would not exist if its tail would not be truncated.
The fact that the measured traffic consists of a mixture of many different types (e.g., web, video streaming, file download), gives an intuition for the encountered heavy-tails of Eq. (5.27) and Eq. (5.30). Accordingly, most sessions are short with low data-volume, consisting of small downloads (e.g., e-mail, TCP-acknowledges), whereas some few sessions last very long and require high throughput (e.g., VoIP, video streaming, file-download). A scatterplot of mean session throughput over session duration is shown in Figure 5.20, in order to visualize the afore-mentioned correlation between both. Again the colors express the common logarithm of the number of samples per square (bin). It is

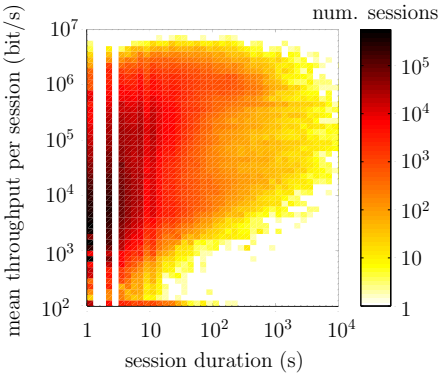


FIGURE 5.20: Scatterplot of mean throughput per user-session over session duration on double-logarithmic scale. The common logarithmic color scale indicates the number of sessions. A strong correlation between both variables is observable.

clearly visible, that sessions with duration of one to three seconds exhibit a throughput concentrated around 10 kbit/s, whereas sessions with durations of 10–100 s show high concentrations at higher throughput (e.g., 1 Mbit/s for session lasting 100 s).

5.3.4 Simulations With Realistic Traffic Patterns

The results presented in the previous section enable the refinement of some simulation scenarios. More precisely, the obtained measurement results question two often made assumptions: (i) the constant number of users – it should be Poisson distributed – and, (ii) the greediness (full buffer) of the users – it should be background traffic, Eq. (5.30). Realistic simulation setups are listed in Table 5.6 and explained in the following.
For the generation of a Poisson distributed number of users, a distinction between snapshot and long-run simulations must be made. For snapshot simulations, the number of users N_u is generated directly from Eq. (5.26). For long-run simulations, on the other hand, it is more convenient to generate user arrivals A_s according to Eq. (5.29) and a respective session duration L_s from Eq. (5.27).
For background traffic simulations, a certain amount of packets must be created for each user. The packets should yield a mean throughput $\bar{R}_s$ according to Eq. (5.30) with packet-arrival times randomly distributed over the simulation time. Estimating statistics for the packet size and packet-arrival times is not possible on the basis of the presented data set, hence, not within the scope of this thesis. Details on that topic are found in [234] [235]. Furthermore, note that the effective mean cell-throughput will differ from the defined mean throughput $\bar{R}_c$, Eq. (5.26), for specific simulation scenarios; the reason being that the correlations between throughput and session duration, see Figure 5.20, are neglected by the proposed models.

Settings Σ, cf. Eq. (5.24)	N_u	A_s	L_s	$\bar{R}_s$
{snap, pois, greedy}	Eq. (5.26)	/	/	max (full buf.)
{long, pois, greedy}	/	Eq. (5.29)	Eq. (5.27)	max (full buf.)
{snap, pois, backgr}	Eq. (5.26)	/	/	Eq. (5.30)
{long, pois, backgr}	/	Eq. (5.29)	Eq. (5.27)	Eq. (5.30)

TABLE 5.6: Different setups for realistic simulation scenarios.

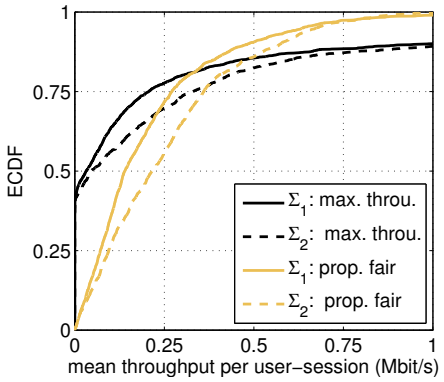


FIGURE 5.21: ECDFs of simulated instantaneous throughput per LTE user. Two schedulers assessed: maximum throughput (dark) and proportional fair (light). Two simulation scenarios: Σ_1={snap, pois, greedy} (solid) and Σ_2={snap, const, greedy} (dashed).

A proof of concept for realistic simulation setups is provided by simulating a Long Term Evolution (LTE) network and assessing the performance of different multiuser-schedulers; namely, the *maximum throughput* and *proportional fair* schedulers. The simulation tool is publicly available at [121], and is explained in more detail in [236] [237]. For the purpose of multiuser scheduling evaluation the proposed {snap, pois, greedy} simulation scenario is suited best. The results are depicted in Figure 5.21 in terms of ECDFs of mean throughput per user-session. Thereby, simulations labeled {snap, const, greedy} assume a constant number of full-buffer users, which is equal to the mean of N_u in the corresponding {snap, pois, greedy} simulation. The figure shows a notable gap between the two simulation scenarios for both schedulers. Note that simulations with the assumption of a constant number of users overestimate the throughput; hence, the more realistic {snap, pois, greedy} simulations should be preferred.

5.3.5 Summary and Criticism

In this section I presented a measurement study on user-behavior observed in a live HSPA network. The data consists of the entire downlink packet-data traffic of approximately 400 cells over a period of three days. Roughly ten million user-sessions are covered. The measurements took place in the city of Vienna, in November 2010.
The main goal was to construct statistical models for HSDPA data traffic in single cells. It shall enable the profiling of cells in an active network. Furthermore, special attention was drawn to the suitability of the developed models for implementation in simulation tools.
Main findings and proposals are:

- The mean throughput of the cells within the peak hour varies over roughly three orders of magnitude. Whereas the 10% of cells with lowest load have a mean throughput of below 1 kbit/s, the 10% most loaded cells have a mean throughput above 500 kbit/s. Consequently, the often made assumption of a constant user density over a large number of cells is inadequate.

- The number of active users per cell is Poisson distributed and a cell can be modeled as $M/G/\infty$ queue. Furthermore, its parameter λ_N shows strong dependence on the average throughput of the cell, Eq. (5.26).

- The mean throughput of a single user-session exhibits a log-normal distribution on exponential scale, Eq. (5.30). The user-session duration fits an exponential distribution on exponential scale Eq. (5.27). Both random processes exhibit heavy-tails, calling for long simulation durations.
- Generative models of user traffic for four different simulation scenarios are given, see Table 5.6.
- Simulations show a relevant difference in performance of multiuser-schedulers when either the models presented in this work or the less realistic assumption of a constant number of greedy users are deployed, see Figure 5.21. The latter tends to overestimate the performance.

The presented study is based on the data set presented in Section 5.3.2, having a temporal resolution of 1 s (due to technical limitations). This resolution inhibits the observation of arrivals of distinct packets or batches of data. Accordingly, no statement can be made on how to model the packet arrivals within a session. Further, the accuracy of the model for the session duration at the lower end is equally impaired by the measurement resolution of 1 s. Moreover, the termination of a session is defined by the lack of transmitted data over the duration of at least one second (timeout). It would be desirable to evaluate various different timeouts, especially shorter ones, which is however not possible due to the measurement resolution.
The targeted cell load, being an input parameter for the modeling framework in Eq. (5.26), does not perfectly match the simulated cell load obtained by {long, pois, backgr} simulations, cf. Table 5.6. The reason is the encountered correlations between $\overline{R_s}$ and L_s shown in Figure 5.20. The accurate modeling of these effects is subject to future work.
The data set concerns randomly picked cells from a bigger area. Thereby, no location information is available. Consequently, it cannot be assessed how strong a specific cell load is correlated to the load of neighboring cells. However, this information would be valuable for the simulation of small wireless networks (e.g., two or three tiers of cells in a honeycomb structure).
Finally, the uplink traffic is not extracted by the measurement software; preventing any statements on relations between uplink and downlink traffic.

Chapter 6

Conclusion

Mobile cellular networks have always been designed and optimized for maximum throughput. The optimization of other parameters has so far been of secondary interest. Due to the involved behavior of the mobile wireless communication channel, those throughput enhancements called for more and more complex transmission schemes; making cellular networks to enormous state-machines with strongly reactive behavior.
Nowadays, mobile networks are able to provide enough bandwidth for most common users. Accordingly, latency will come into vogue as optimization parameter for enhancing the user satisfaction. A respective example is the LOLA project, for which most of the work of this thesis was aimed.
This thesis tries to give a complete and self-contained tutorial on latency measurements. Although, the focus is on mobile wireless networks (including several details which may not apply for any other type of network), most discussions are suitable for latency measurements in general. The covered aspects are:

- the definition of latency,
- network models required for the measurement design,
- measurement design methodologies,
- timekeeping,
- measurement precision,
- packet injection and tracing,
- measurement hardware,
- fair benchmarking of networks,
- the influence of traffic patterns on delay and
- accurate modeling of various traffic patterns.

6.1 Lessons Learned

During the past years, I gained several insights into the present topic, the most important are:

- Literature provides several different definitions of delay, which depend on (i) the measurement hardware, (ii) the network model and (iii) the field of application of the respective results. The practitioner shall choose wisely which definition to adopt.
- Delay in mobile cellular networks depends on a multitude of factors. This is due to the diversity of the involved communication links (wireless links as well as backbone links). A measurement design shall aim to handle (either resolve or eliminate) all of these factors.
- Mobile cellular networks are reactive; meaning that the injected traffic pattern influences the experienced latency. The well-known delay measurement techniques are not able to handle this reactiveness.
- Several aspects of the traffic pattern can cause reactions of the network. The presented measurements provide evidence that such reactions depend on several statistical measures (i.e., distributions, auto and cross-correlations) of packet sizes and inter packet-arrival times, respectively. Therefore, any measured delay figure is only valid for traffic patterns which resemble the measurement pattern in the above regards.
- In case synthetic traffic is used for measurements, a precise traffic model is vital for the correct assessment of latency figures. Common simple traffic models are not satisfactory for this purpose.
- Benchmarking (comparing and ranking) reactive networks is a non-trivial task, since different traffic patterns potentially yield different rankings. Consequently, benchmarking techniques should include measurements from a variety of different traffic patterns in order to have general validity.

Several secondary aspects covered within this thesis, such as clock synchronization and traffic modeling, are interdisciplinary and contain beneficial information for several other applications.

6.2 Outlook

Since the journey to low-latency cellular networks began only recently, I expect a strong growth of interest in this topic in the future. Beside the currently available real-time applications, several new applications which require low latency will appear in the future. Current buzz words are, for example, Machine-to-Machine Communication (M2M) and smart grid; the respective deployment over cellular networks will impose heavy challenges on the network design – especially in terms of latency.
In turn, network providers can create new sources of revenues by guaranteeing maximum latencies for packet delivery. Such scenarios will require reliable and standardized latency measurement procedures, in order to enable the customer to evaluate the adherence to the quality of service agreements. Consequently, further research on delay benchmarking strategies is required and will appear, in order to satisfy future needs.

Appendices

Appendix A

Protocol Descriptions

A.1 Internet Protocol Version 4

The Internet Protocol (IP) enables the delivery of datagrams to well-defined destination hosts over multiple networks. Its purpose is packet routing beyond the network borders, thus, it allows for inter–networking. It is defined in Request For Comment (RFC) 791 introduced by the Internet Engineering Task Force (IETF) in the early 1980s [238]. IP is a connectionless protocol or, equivalently, working in packet-switched operation. Consequently, transmitted packets are not restricted to follow a fixed reliable path to their destination; they may be sent over multiple routes, be duplicated or lost. IP introduces three main functionalities in order to fulfill its purpose: (i) unambiguous host addressing, (ii) payload fragmentation and (iii) definition of the lifetime of datagrams. An IP datagram consists of one or more fragments, each of which consisting of a header field with 20 B or more and the data field with a maximum of 65 515 B. The frame structure of an IP fragment is depicted in Figure A.1. It consists of the following fields:

- Version: This field consists of 4 bit, indicating the version of the IP protocol in use. In the present case this number is always equal to four.
- IHL (Internet Protocol Header Length): This field contains a 4 bit number indicating the length of the header field. This field is required for the header options, which have variable size. The length is given in multiples of 4 B; hence, for the standard header field (without options) the number is 5, whereas the maximum header length equals 15×4 B=60 B.

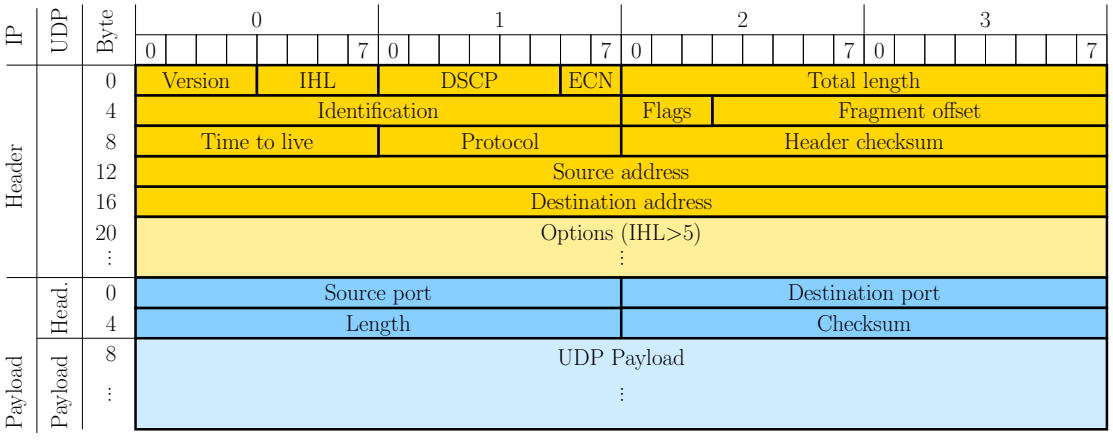


Figure A.1: Packet format of an IP datagram with embedded UDP payload.

- DSCP (Differentiated Service Code Point): Originally intended as service type field, it is now specified in RFC 2474 as a possibility for quality-of-service differentiation (DiffServ). The feature requires support of lower layers.
- ECN (Explicit Congestion Notification): Specified in RFC 3168, this value allows for the end-to-end notification of network congestion and is intended to reduce packet loss in such cases. However, the feature is optional and requires support from both endpoints, as well as the underlying layers to be effective.
- Total length: This 16 bit number indicates the total length of the present IP fragment in bytes. Thereby, the header length is included. Accordingly, the maximum value is 65 535 B, the minimum 20 B.
- Identification: This field is required for the fast and unique identification of the IP fragment.
- Flags: The three included flags concern payload fragmentation and are the following: (i) reserved, (ii) don't fragment, (iii) more fragments. The first one is always zero, the second indicates that packets shall rather be dropped than segmented and the third suggests that the IP datagram consists of more fragments (the present one is not the last).
- Fragment offset: This 13 bit number specifies the offset of the beginning of the present fragment relative to the beginning of the payload. The respective length is given in multiples of 8 B, in order to allow for an offset up to the maximum payload length of 2^{16} B. Accordingly, the offset of the first fragment is zero.
- Time to live: Ideally, this field specifies the number of seconds until the present IP fragment becomes invalid. In practice, however, this field is a hop-count, with each router on the path decrementing the respective value. Accordingly, this field prevents packets from circulating in loops of the network.
- Protocol: This field indicates the layer 4 protocol of the IP payload. The respective numbers are defined in RFC 790.
- Header checksum: This value is a 16 bit checksum calculated from the header, in order to guarantee its respective integrity. It is specified in RFC 1071. When a checksum indicates a corrupted packet, it is dropped immediately. Routers, which modify the IP header have to calculate the new checksum.
- Source address: This 32 bit value identifies the host being the sender of the packet. For more information on this topic refer to [14].
- Destination address: This 32 bit value identifies the host being the receiver of the packet.
- Options: This field is optional and rarely used.

The IP protocol supports payload fragmentation, as a service for lower layers. Layer 2 protocols may only be able to handle batches of payload smaller than a given size. This maximum size is called Maximum Transmission Unit (MTU) and may be different for each link along the data path. Therefore the IP payload may be packed into multiple IP fragments, each with size smaller than the MTU. Fragmentation may also be performed in layer 3 routers along the data path, since the source host is not aware of all MTUs of

the route. Reassembly however, is only performed at the destination host. The reason is that packets may travel on different routes to their destination, yielding intermediate reassembly unfeasible.

A.2 User Datagram Protocol

Layer 4 communication protocols (transport layer) provide features for the transparent data delivery between processes on remote hosts. Those can be: (i) process-to-process delivery, (ii) error detection, (iii) reliable or error-free transmission, (iv) connection oriented transmission (v) segmentation and reassembly and (vi) flow control. The UDP protocol, specified in RFC 768 in 1980 [239], is one of the most basic transport protocols and provides only the first two of the mentioned features; therefore it is termed unreliable, connectionless and stateless. It enables the exchange of data between processes on remote hosts, although, there are no mechanisms to assure that the data arrived (no handshaking). Further, there may be duplicated datagrams. Nevertheless, UDP provides integrity checking of the received data (error detection).
The process-to-process delivery is enabled by so called port numbers. Those are 16-bit numbers which uniquely define the source and destination processes. The lowest ports (0–1 023) are well-known ports, dedicated to specific services; their use may be handled by the host itself (operational system), especially if the host is a server. Ports 1 024–49 151 are registered ports, commonly used by popular applications. The higher ports (49 152–65 535) are called ephemeral ports and can be used dynamically by any arbitrary application.
The UDP datagrams consist of the unmodified payload and a header of 8 B. The maximum length of the payload is 65 527 B; however, this payload length may be limited by the underlying layer 3 protocol. In the case of Internet Protocol Version 4 (IPv4) for example, the maximum IP datagram length is 65 535 B, yielding a maximum UDP-payload length of 65 507 B after subtracting the minimum header length of both protocols. The frame structure is depicted in Figure A.1. The UDP header contains the following fields:

- Source port: This number identifies the source process on the source host. It is used for any reply.
- Destination port: This number identifies the destination process on the destination host.
- Length: This field specifies the length of the entire UDP datagram (header and payload) in bytes.
- Checksum: It enables the examination of the integrity of the datagram. The respective calculation, outlined in RFC 768, is based on the entire datagram with a pseudo header. Beside of the UDP header, the pseudo header includes information of the IP header, which enables the detection of an erroneous host-delivery by layer 3.

Appendix B

Influence of the Traffic History in Queueing Theory

Reactive effects, as outlined in Section 2.3.2, are not the only cause for influences from the history of the traffic pattern on the currently experienced delay. Such effects could also be caused by pure queueing. In the following I analyze the likelihood of this case. Therefore I consider only the worst traffic patterns deployed in Section 4.1 and Appendix D. It turns out that the probability that the history of a traffic pattern influences the presently experienced delay is negligible, provided that the cross-traffic is light (i.e., $U \leq 0.3$).

B.1 Suppressed Influence

As mentioned in Definition 2.5, the reactiveness of a network node or route section can only be verified for low network load. This does not mean that reactiveness vanishes at high work loads, but queueing of packets itself causes the experienced delay to depend on previous packets. This is due to the backlog of the queue, which may be substantial for high utilization (workload). Considering the arrival instant of the current packet, the queue is longer as it would have been without the previous probing packets.
This effect is suppressed when the queue is **emptied between each pair of probing packets**. An illustration is given in Figure B.1, which shows the temporal evolution of the buffer length (queue size) for two cases: (a) without probing packets and (b) with the largest possible probing packets. Thereby, the interval τ between the two probing instances corresponds to the minimum possible inter packet-arrival time. Since the queue is empty at the marked instances for both cases, the presence of the probing pattern does not change the queue size at the instance of next arrival. In other words,

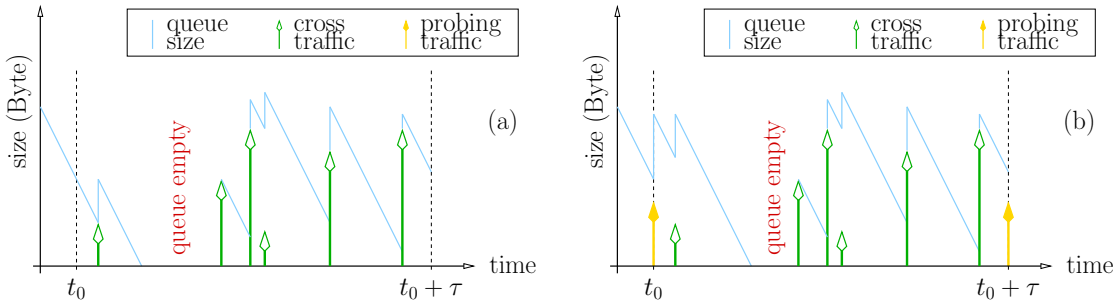


Figure B.1: Influence of the history of the probing pattern for low queue utilization. (a) queue length without probing, (b) queue length with probing. Note, that both queue sizes are coinciding at $t = t_0 + \tau$.

any possible probing pattern would have caused exactly the same queue size at $t{=}t_0{+}\tau$. The influence of the history of the traffic pattern is therefore canceled.

B.2 Example: Worst Case Scenario

To quantify this cancellation effect, I provide an example, considering the worst case scenario which can be deduced from the settings in Section 4.1 and Appendix D. This example yields an upper bound on the utilization (or maximum rate) of the cross-traffic. When the cross-traffic is below this bound, then any influence from the history of the traffic pattern on the actual delay is very likely due to *reactiveness* instead of queue backlog. The following assumptions are made:

- the network capacity of $C_{\max}{=}1\,\mathrm{MB/s}$ (reasonable for modern networks),
- a probing rate of $R_p{=}100\,\mathrm{kB/s}$ (maximum in Appendix D), yielding $U_p{=}0.1$,
- a inter-arrival time $\tau{=}10\,\mathrm{ms}$ (minimum value deployed in Appendix D),
- cross-traffic rate $R_x{=}U_xC_{\max}$ with utilization U_x (manipulated variable) and
- an $M/D/1$ queue, with Poisson arrivals and constant serving time (the latter determined by the maximum MTU of the cross-traffic: $\pi_x{=}1500\,\mathrm{B}$).

The overall utilization U is split into two contributions: $U{=}U_p{+}U_x$, where U_p denotes the utilization due to probing traffic and U_x the utilization due to cross-traffic.
In order to simplify the evaluation, the notion of *virtual* queue size at t_0 is introduced; being an upper bound on the real queue length at this time instant. The *virtual* queue size at t_0 is composed by the sum of the following components:

- the actual queue size right before t_0 (arising from packet arrivals before t_0),
- the size of the probing packet $\pi_p{=}\tau R_p$ and
- the volume of the cross-traffic arriving within the future interval $[t_0, t_0{+}\tau]$.

The probability P_{NE} that the queue does not run dry during the interval $[t_0, t_0{+}\tau]$ is upper bounded by the probability that the virtual queue size at t_0 is higher than the amount of data which can be processed within the interval (i.e., $\tau \cdot C_{\max}$). Equivalently, this can be formulated as

$$P_{NE} = P\left\{N_0 + N_\tau \geq \tfrac{\tau C_{\max} - \pi_p}{\pi_x}\right\}, \tag{B.1}$$

where N_0 denotes the number of packets in the queue at time t_0 and N_τ is the number of packets arriving in the interval $[t_0, t_0{+}\tau]$. The distribution of N_0 can be determined by the *Pollaczek-Khinchin transform*, cf. [24, p. 313, Eq. (7.66)], to

$$P\{N_0{=}n\} = (1-U)\sum_{k=0}^{n} e^{kU}(-1)^{n-k}\tfrac{(kU+n-k)(kU)^{n-k-1}}{(n-k)!}. \tag{B.2}$$

The number of future arrivals follow a Poisson distribution $N_\tau \sim \mathcal{P}ois(\lambda_x)$ (cf. Appendix H) with rate

$$\lambda_x = \tfrac{\tau C_{\max} U_x}{\pi_x}. \tag{B.3}$$

Both variables are independent, since the arrival process is Poisson (i.e., past arrival are independent of future arrivals at each point in time, cf. Section 2.3.1.2). Therefore, the

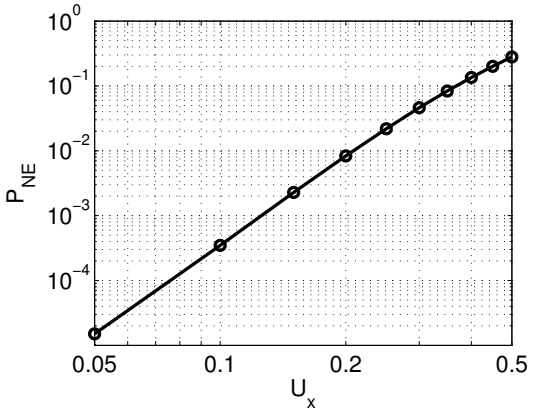


FIGURE B.2: Upper bound on the probability P_{NE} of the event that the queue was never empty between two probing packets over the utilization due to cross-traffic U_x.

Probability Mass Function (PMF) of the respective sum corresponds to the convolution of the individual PMFs. Accordingly, Eq. (B.1) can be evaluated in closed form.
The result is depicted in Figure B.2, which shows P_{NE} over the utilization of the cross-traffic U_x. The probability that the queue does not run dry within $[t_0, t_0+\tau]$ is negligible for low utilizations. For cross-traffic utilizations of $U_x \leq 0.2$ the probability $P_{NE} \leq 1\%$. Equivalently, only 1% of the delay values is influenced by previous packets. This is considered as sufficient for the detection of reactiveness. Notice, that the displayed curve is an upper bound on the worst case scenario. Changing the interval between probes to τ=100 ms (but leaving R_p unchanged) would, for example, yield $P_{NE}=10^{-20}$ for U_x=0.2.
Summarizing, light cross-traffic combined with light probing traffic (overall utilization $U \leq 0.3$) ensures that *reactiveness* of network nodes or route sections can reliably be detected. This assertion holds for all relevant scenarios outlined in Section 4.1 and Appendix D. Network providers usually install over-dimensioned links, especially at the air interface of cellular networks. Further, the network load strongly varies over the day; e.g., by a factor of ten (as outlined in Section 5.3, Figure 5.16). Consequently, performing measurements at off-peak hours is very likely to results in favorable conditions for the detection of reactive effects.

Appendix C

Timekeeping on Desktop PCs

There are different ways to compensate for clock instabilities of PCs as discussed in Section 3.1. Most operational systems provide ways to constantly adjust the software clock to remote servers in the Internet. Prominently, the Network Time Protocol (NTP) [81] is used for distributing timing information in such systems. Remote timeservers synchronized to the Coordinated Universal Time (UTC) transmit timestamps to PCs on which sophisticated algorithms adjust the software clock in regular intervals. A synchronization accuracy of 100 ms down to 1 ms can be achieved, depending on the network path to the timeserver. A more accurate solution (1 μs–100 μs) is provided by the IEEE 1588 standard [88], which is used for distribution of timing information over local networks. However, if synchronization to UTC is needed, the master device is responsible for providing respective timing information. Furthermore, the Global Positioning System (GPS) provides UTC timing information with accuracy of below 100 ns [79]. The drawback is that the receiving device requires a clear view to the sky. The cost of GPS receivers has become very low (below EUR 50) in recent years, yielding the respective solution an attractive alternative to the above described methods.
In this appendix I discuss the different possibilities for low-cost time synchronization of desktop computers. For their evaluation, I propose a novel measurement setup, capable of precisely measuring the timing offset of the software clock, by using a rubidium frequency standard synchronized to UTC. Moreover, extensive measurement results are provided regarding the accuracy of synchronization of *Linux* machines; i.e., for different PC architectures, different interface cards, different load scenarios and different timekeeping software. An analysis of the transient components of the synchronization process is provided. Finally, fundamental limitations of timekeeping on desktop PCs are identified; for example, the interrupt latency.

C.1 Synchronization Techniques

Although crystal oscillators can achieve very good performance in terms of frequency stability [79], those in standard desktop computers are not sufficiently dimensioned for precise timekeeping. An example is shown in Figure C.1 (free running PC). In this case the drift (skew, first derivative of the offset, cf. Section 3.1) of the free running PC clock is in the order of 10^{-7}, a quite reasonable value. Nevertheless, a drift of up to 10^{-4} was observed on the same machine, depending on the temperature of the environment. The fact that the CPU load strongly influences this temperature is particularly inconvenient. A positive property of the PC oscillator is the high short term stability of the skew (10^{-7} or better), which allows for notable improvements in stability by continuously compensating for the drift.

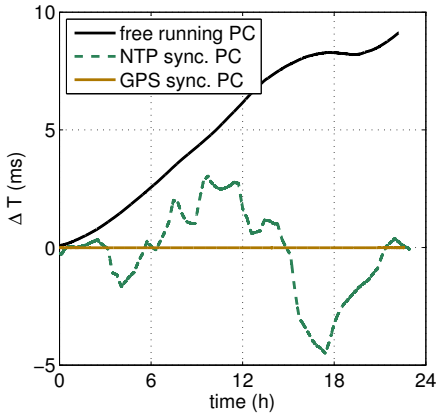


FIGURE C.1: Clock offset to UTC over time for different synchronization techniques over 24h. The PC runs *Linux*.

The probably most often deployed approach for time synchronization is using dedicated software for the Network Time Protocol (NTP) [80] [240]. Nowadays there are thousands of servers in the Internet, distributing timing information. Client software on the local machine executes sophisticated signal processing algorithms such as frequency and phase locked loops and adjusts the software clock. Those clients are often part of the operational system. One of the most popular is *ntpd* [240], which, beside of handling NTP, is also able to handle other synchronization techniques, such as GPS or atomic clocks. The achievable accuracy using NTP is in the range of 1 ms–100 ms, measured and evaluated in several works [241] [242]. A comparison between *Windows* and *Linux* operating systems is given in [243]. A sample of the temporal behavior of a software clock synchronized by NTP (by using the *ntpd* tool) is presented in Figure C.1. Three timeserver located in Austria and mediated by *pool.ntp.org*, have been used for clock adjustment.

Timing devices with an accuracy of below 1 μs (such as GPS receivers, long-wave radio time signal receivers or atomic clocks) often provide a Pulse Per Second (PPS) output signal, a pulse with some microseconds duration transmitted every second. Unfortunately, hardly any common operational system provides native interfaces for these timing devices. Therefore, NTP servers often use dedicated hardware, but there also exist software solutions for synchronizing *FreeBSD* and *Linux* desktop PCs to PPS signals. An example for a time series of the synchronization offset of such a solution is

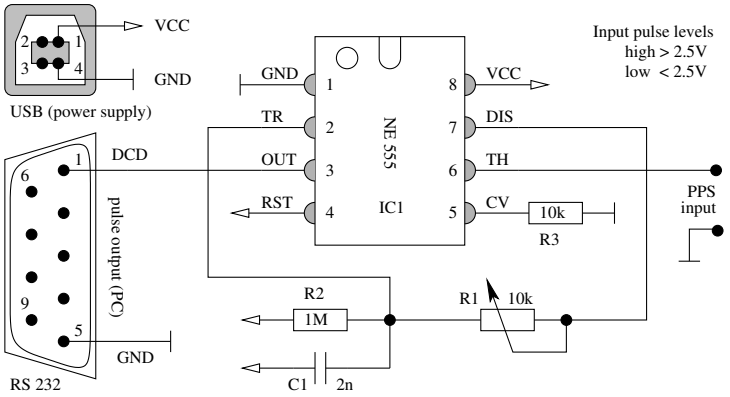


FIGURE C.2: Level conversion circuit for connecting a timing device to the serial port (RS232) of the PC. Includes monoflop for pulse enlargement.

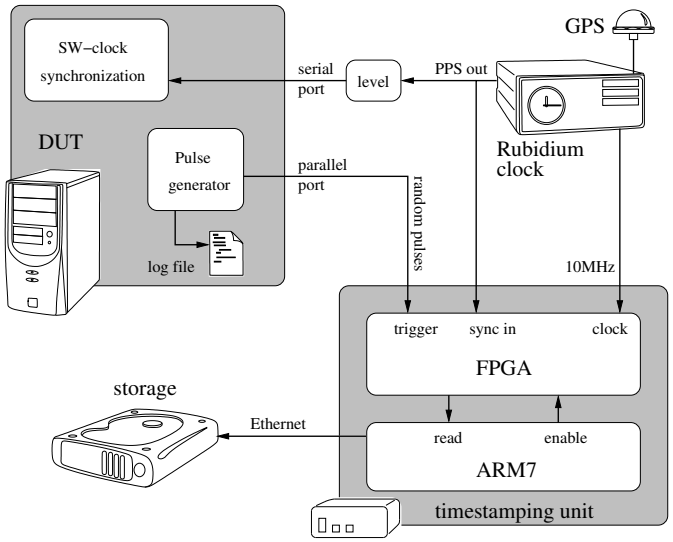


FIGURE C.3: Setup for synchronization precision measurements.

shown in Figure C.1 (GPS synchronized PC, continuous flat line). For the rest of this work the synchronization of *Linux* machines is performed by using two variants of PPS compatible drivers: (i) *gpsd* [244] and (ii) *LinuxPPS* [93].
The first driver is a user space daemon providing communication with several GPS receivers. The capability of handling synchronization pulses (which have to be timestamped) is an additional feature. The setup consists of standard versions of *gpsd* and *ntpd*, configured to communicate over shared memory access. Installation needs no modification of the kernel and even installation from repositories is possible. To run the timestamping instance *gpsd* in user space domain is detrimental to clock synchronization accuracy, since execution depends on the system scheduler. Under heavy CPU load conditions or frequent interrupts, the performance will deteriorate, see Section C.3.
The second driver, *LinuxPPS*, is a kernel driver. It is especially designed for handling PPS input signals; hence, timestamping of these pulses is done in kernel space. Modules for *ntpd* able to interact with the driver are available but require compilation of *ntpd*. Since kernel version 2.6.34, the *LinuxPPS* code has been merged into the *Linux* source tree; thus, the driver can be used directly without patching the kernel. The execution in kernel domain provides good performance, even under load (see Section C.3).
Both drivers described above expect the PPS signal connected to the DCD pin of the RS232 interface (serial port) of the PC. Since most of the commercially available precision time sources are not compatible to RS232 voltage levels and the pulse duration may be too short, a level conversion circuit is needed (costs: roughly EUR 5), see Figure C.2. It creates an extra delay of roughly 250 ns.
Efforts implementing synchronization of *Linux* PCs deploying IEEE 1588 have started only recently. Although very good performance can be achieved on dedicated hardware, the available software solutions (e.g., *ptpd* [90]) are expected to perform worse than *LinuxPPS*. This is due to the relatively high delay caused by the network interface card [245]. Furthermore, IEEE 1588 does not provide mechanisms for absolute synchronization to UTC. For those reasons this technology is not considered for further investigations.

C.2 Measurement Setup

In order to measure the offset of the software clock compared to UTC the measurement setup presented in Figure C.3 is deployed. The main timing source is a rubidium clock (SRS FS725), with a short term stability of 10^{-11} (assessed over a 1 s interval). It is synchronized to a GPS receiver for long term stability. The timing information delivered by this device is used to synchronize, both the Device Under Test (DUT) and the timestamping unit.
A custom timestamping unit [246], designed for precise handling of timing information is incorporated into the measurement setup. It comprises a Field Programmable Gate Array (FPGA) (Xilinx XC2C512) and an ARM7 microcomputer (NXP LPC2368). The FPGA provides stopwatch functionality, deploying synchronous hardware counters and is running directly on a harmonic of the 10 MHz frequency output of the rubidium clock. The stopwatch is started by a pulse at the *synchronization* input (provided presence of the enable signal from the controller) and interim times are taken at the rising edge of pulses at the *trigger* input. Beside of controlling the stopwatch, the microcontroller is reading those interim timestamps. Furthermore, it provides an Ethernet interface, through which it sends the data to an external storage for later evaluation. The timestamping accuracy of the unit is estimated to be within 200 ns to UTC.
The Device Under Test (DUT) is synchronized to UTC using the PPS output of the rubidium clock adjusted by the level conversion circuit (cf. Figure C.2). Additional means of synchronization are feasible (e.g., NTP over Ethernet connection for determining the absolute UTC second), as well as all other synchronization techniques described in Section C.1. An extra program runs on the DUT, generating short pulses at its LPT port (parallel port), with random inter-arrival time and a frequency of approximately 15 pulses/s. The command used for this purpose is `outb()`, which is blocking and able to generate pulses with duration of 1–2 μs; hence, I estimate the extra delay to be situated in the same range. The pulses cause a trigger event at the timestamping unit, resulting in interim timestamps, which the microcontroller stores on the storage device. Moreover, the time of the rising edge of the pulses is recorded and stored in a log file on the DUT. For this purpose, timestamps are acquired before and after the edge, deploying the `clock_gettime()` command, whose execution lasts approximately 400 ns, depending on the hardware performance of the DUT.
The measurement process is depicted in Figure C.4. The clock of the timestamping unit is derived directly from the rubidium clock, the introduced delay is assumed negligible. When the microcomputer enables the stopwatch, the next full UTC second causes it to start. Each second synchronization pulses are send from the rubidium clock to the DUT. The critical part of this procedure is the timestamping of the synchronization pulses at the DUT. This is achieved by an interrupt which reads and stores the actual time of the software clock, its duration is $\delta_{\text{in}}[n]$, see Figure C.4. According to these time values the internal clock is adjusted, deploying the frequency and phase locked loops provided by the kernel, what however is not time-critical. As a consequence the control loop causes the software clock T_{sw} to converge not to UTC but to an earlier time,

$$T_{\text{sw}} = T_{\text{UTC}} - \delta_{\text{in}}[n]. \tag{C.1}$$

Hence, when measurement pulses are issued by the DUT, the internally recorded timestamps are not the real UTC timestamps but these earlier. Arrival of the pulses at the timestamping unit causes the stopwatch to output and store the real UTC timestamp on the external storage. Comparing both internal and external timestamps the resulting difference $\delta_{\text{sum}}[n]$ has two reasons, namely, first the duration of the processing

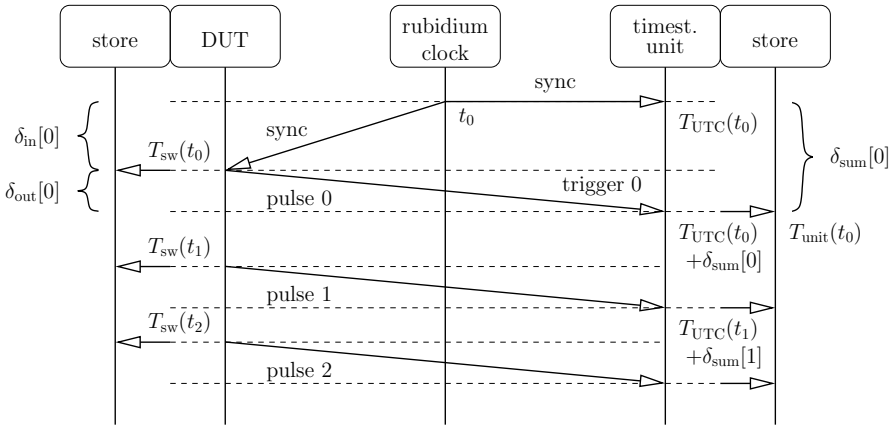


FIGURE C.4: Flowchart of the measurement process.

of the synchronization pulses from the rubidium clock by the PC ($\delta_{\mathrm{in}}[n]$) and second the duration of the output procedure of the pulses excited by the DUT ($\delta_{\mathrm{out}}[n]$),

$$\begin{aligned}\delta_{\mathrm{sum}}[n] &= T_{\mathrm{unit}}(t_n) - T_{\mathrm{sw}}(t_n) \\ &= \Big(T_{\mathrm{UTC}}(t_n) + \delta_{\mathrm{out}}[n]\Big) - \Big(T_{\mathrm{UTC}}(t_n) - \delta_{\mathrm{in}}[n]\Big) \\ &= \delta_{\mathrm{out}}[n] + \delta_{\mathrm{in}}[n]. \end{aligned} \tag{C.2}$$

Delaying factors such as scheduling of the operating system can be minimized or eliminated by choosing the start time of DUT induced pulses completely random. Effectively, the duration of this second procedure ($\delta_{\mathrm{out}}[n]$) boils down to the duration of the consecutive execution of the `clock_gettime()` and the `outb()` commands. I expect this method to outperform the more common way of outputting timestamps on the Ethernet interface [92] [83], the reason being that the issued software commands are less complex and less hardware components are involved in the procedure (processor, north-bridge and south-bride compared to processor, north-bridge, south-bridge and network-interface card). The interval $\delta_{\mathrm{out}}[n]$ is shorter than $2\,\mu$s and $\delta_{\mathrm{sum}}[n]$ is considered as a worst-case approximation of $\delta_{\mathrm{in}}[n]$. The fact that the frequency stability of the crystal oscillator of the DUT is very high on short timescales (10^{-7}), allows for condensing all $\delta_{\mathrm{sum}}[n]$ recorded within the same UTC second (roughly 15) to their minimum value, since their variation must descend from $\delta_{\mathrm{out}}[n]$. Causality suggests the minimum $\delta_{\mathrm{sum}}[n]$ as best approximation for $\delta_{\mathrm{in}}[n]$. This policy further improves the estimation of $\delta_{\mathrm{in}}[n]$ and yields equidistant time series which are easy to evaluate. For the rest of this document such estimates are referred to as ΔT.

C.3 Results

All results presented in this section base on 24 h measurement runs performed with the setup presented in Section C.2. Figure C.5 (a) shows ECDFs of the misalignment of the software clocks for different computers compared to UTC. The processors, chipset or mainboard types and year of purchase are listed in Table C.1. For synchronization *ntpd* was deployed in combination with *LinuxPPS*, whereas additionally connections to public NTP servers have been established to detect the absolute UTC second. It is visible, that all PCs suffer from a synchronization offset with non-zero mean (timing

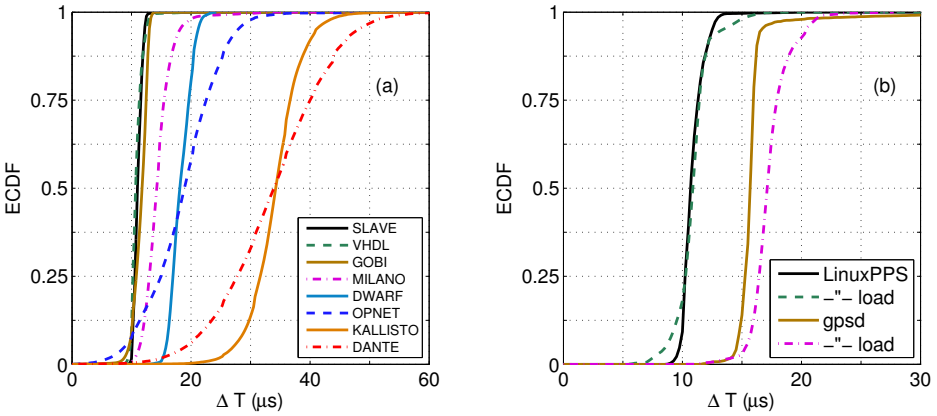


FIGURE C.5: (a): ECDFs of synchronization accuracy over 24 h, different PCs. Synchronized by *ntpd* with *LinuxPPS*. (b): ECDFs of synchronization accuracy under load over 24 h. Host: VHDL, different drivers for *ntpd*.

lag), from 11.4 μs (on SLAVE) up to 34.2 μs (on DANTE). This is due to the interrupt latency ($\delta_{\text{in}}[n]$) as pointed out in Section C.2. For a deeper insight into the interrupt process on modern PCs and the corresponding latency see [247], from which it becomes clear that the chipset (i.e., memory controller hub, I/O controller hub) strongly influence interrupt lags. Moreover, the observed standard deviations of the timing errors are an order of magnitude below the respective lags, namely, between 0.65 μs (SLAVE) and 10.8 μs (DANTE). *ntpd* provides the option to compensate for fixed lags, which is highly advantageous, especially in the case of computers with higher performance. If such compensation would be applied to SLAVE, VHDL or GOBI, those machines would yield a precision in the microsecond range. Unfortunately, the only general presumptions about mean lag, which holds for different computer platforms, is to assume it positive. Hence, considering compensation, the interrupt latency of each platform has to be evaluated separately.

A pragmatic and inaccurate, though practical approach for such an evaluation as alternative to the costly measurement setup in Figure 3.1 may be performed using the circuit proposed in Figure C.2. *LinuxPPS* detects both rising and falling edge of the PPS signal. They are timestamped in a blocking interrupt routine. If the PPS pulse is short enough, the falling edge will not be detected, since the interrupt routine caused by the rising edge is still in execution and will prevent any other interrupt. Using the circuit shown in Figure C.2, the duration of the PPS pulse can be varied between 1–50 μs by adjusting potentiometer R1. Consequently, one may start with pulses of 1 μs duration and slowly

Name	Processor	Mainboard / Chipset	Year
DWARF	Intel Atom 330	Intel 945GC / ICH7	2010
DANTE	AMD AthlonXP 1800+	VIA Apollo KT266A	2002
GOBI	Intel Core2 Quad 2,4	Asus P5B-VMSE	2009
KALLISTO	Intel Core2 Duo 1,83	Asus P5L-MX	2007
MILANO	AMD Athlon64 3200+	Asus A8V	2006
OPNET	AMD AthlonXP 2800+	Asus A7V-600X	2004
SLAVE	Intel Core2 Duo 2,66	Asus P5B	2008
VHDL	Intel Core2 Duo 1,86	Asus P5B	2009

TABLE C.1: Analyzed computers and architectures.

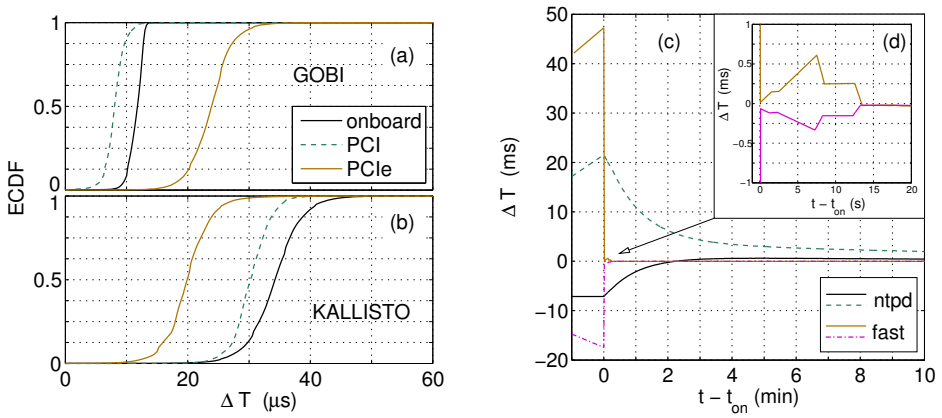


FIGURE C.6: (a–b): ECDFs of synchronization accuracy over 24 h, different interface cards. (c–d): Transient analysis of the synchronization process, different software.

increase the duration until interrupts caused by the falling edge can be observed. The resulting duration is a rough estimate for the lag $\delta_{\text{in}}[n]$ and can be consigned to *ntpd* for compensation.

Further measurements have been conducted considering variations of CPU load, see Figure C.5 (b). Thereby the performance of the drivers *gpsd* and *LinuxPPS* are compared, whereas *ntpd* was used as timekeeping software for both experiments. To emulate a loaded scenario, a *Perl* script is executed which performs arithmetic operations in an endless loop, pushing CPU utilization to 100 %. Furthermore, it also produces slight network load, in order to cause scarce interrupt activity. The measurements have been performed on VHDL. Two main effects can be observed in the resulting figure. First, *LinuxPPS* performs better than *gpsd* in unloaded scenarios (solid lines). This behavior results from the fact that one is integrated in the kernel of the operational system, the other is executed from user space. Second, the performance of *LinuxPPS* decreases only marginally under load (dashed line), whereas the performance of *gpsd* is noticeably impaired (dotdashed line). This effect may be caused by the lower priority of user space programs compared to kernel routines. Nevertheless, the impact of high CPU load is minor and both driver perform reasonably well.

Less consistent outcomes are obtained from measurements using different RS 232 interfaces on the same machine. Two hosts, GOBI and KALLISTO, are used for measurements deploying onboard RS 232, PCI and PCIe extension cards. Results are shown in Figure C.6 (a–b). Whereas for GOBI (a) the PCI interface performed best and the PCIe interface worst, the situation is totally different for KALLISTO (b), where PCIe performed best and onboard worst. The resulting distributions strongly vary in mean lag and variance. I conclude that the mainboard design and especially the involved chipsets heavily influence the synchronization properties [247].

An analysis of the transient components of the synchronization process is shown in Figure C.6 (c). Four measurement curves are shown, namely, two resulting from synchronization deploying *ntpd* (with positive and negative initial offset) and other two representing transient components of the synchronization using a custom timekeeping software, I designed for fast convergence to UTC. This program, in the following named *fast*, can be downloaded at [121]. The value zero of the x-axis corresponds to the instant the synchronization mechanism is turned on. Before this point in time the clocks are running free. All setups deploy *LinuxPPS* for timestamp acquisition. It is clearly visible from the figure, that *ntpd* takes several minutes to converge to the region of ± 1 ms to

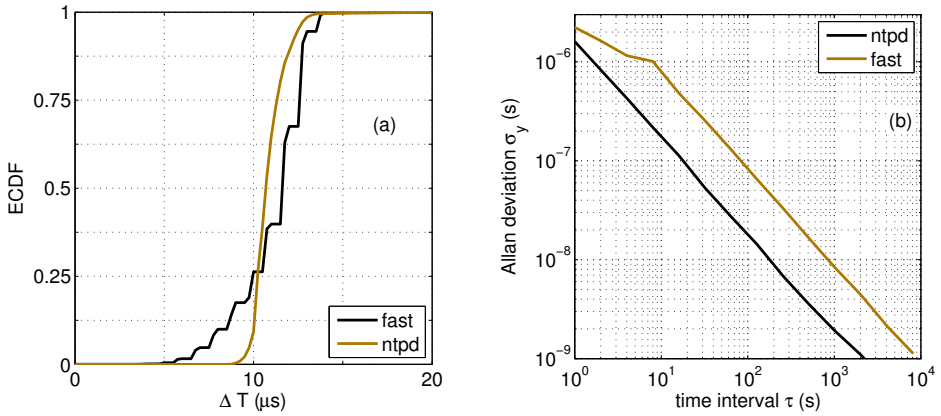


FIGURE C.7: (a): ECDFs of synchronization accuracy over 24 h, different software. (b): Allan deviation of PC clock, different software, measured on GOBI.

UTC (note that the units on the ordinate of Figure C.6 (c) are milliseconds). Furthermore, the controller may cause overshoots, as for the solid dark curve. Depending on the initial value of time difference to UTC (ΔT), it may take one to several hours until *ntpd* converges to the region of 0–50 μs to UTC. This behavior is disadvantageous for certain applications, for example, where frequent reboots or unreliable connection to the Internet are expected [1].

On the other hand *fast* is converging instantly to the desired region, as demonstrated in the respective display detail in Figure C.6 (c). *fast* consists of a state machine with three states, the first one uses *ntpdate* to bring the PCs software clock within ± 0.5 s to UTC, afterwards the second uses the `settimeofday()` function to reduce the offset to UTC to zero. The third state, responsible for fine tuning, estimates the frequency offset and to compensate for it. A median filter of 5 s length estimates the offset and the `adjtime()` function is afterwards used to compensate for it. On top of this, another control loop adjusts accumulated phase errors. The transient behavior is exemplified in the display detail of Figure C.6 (c). In the interval before second 3 the software timer is adjusted to UTC (first two states). Between second 3 and 8 the frequency offset is estimated for the first time. In the next five seconds the frequency offset is strongly reduced, observable at the slope of the curve, which is close to zero. In second 13 compensation for the accumulated phase error takes place, what brings the software timer rather close to UTC. Hence, the transient synchronization process, leading within the region of 0–50 μs to UTC, takes far less than one minute. Interestingly, the synchronization quality in terms of difference to UTC does not suffer substantially compared to *ntpd*. This is pointed out in Figure C.7 (a).

One significant drawback of *fast* compared to *ntpd* is its relatively low stability. *ntpd* deploys advanced signal processing algorithms to guarantee stability, including preceding samples, whereas *fast* starts from scratch each 5 s. A common way to measure oscillator stability is the Allan deviation $\sigma_y(\tau)$ for the measurement interval τ [78]. It describes the unpredictability of an offset-sample x_k given its two predecessors x_{k-1} and x_{k-2}. By use of the fractional frequency

$$y_k = (x_k - x_{k-1})/\tau \tag{C.3}$$

the Allan variance is defined as

$$\sigma_y^2(\tau) = E\{(y_k - y_{k-1})^2\} \tag{C.4}$$

where $E\{\}$ is the expectation and can be estimated by

$$\hat{\sigma}_y^2(\tau) = \frac{1}{2(N-2)\tau^2} \sum_{k=2}^{N-1} (x_k - 2x_{k-1} + x_{k-2})^2. \tag{C.5}$$

where N denotes the number of samples. A respective evaluation of both *ntpd* and *fast* synchronization programs is given in Figure C.7 (b). As observable, *ntpd* is able to keep the slope of σ_y at -1, whereas *fast* shows a plateau around 5 s, which is exactly the basic heartbeat of its fine-tuning mechanism. Losses of factor five are experienced for longer τ.

C.4 Summary

In this work I present a method of measuring the time offset of software clocks in desktop computers with respect to UTC. The precision of the method is roughly within $\pm$ 1 μs, depending on the performance of the DUT. Furthermore, an evaluation of different synchronization techniques for commodity computers is shown. The following parameters have been analyzed: (i) the hardware architecture, (ii) the CPU load, (iii) the interface on which the synchronization pulses are fed to the kernel, (iv) the driver used to read the synchronization pulses and (v) the timekeeping software itself. Different types of analysis are presented, namely, the time series of the offset of the software timer, Cumulative Distribution Function (CDF) plots of the offsets and Allan-variance plots concerning stability of the software oscillator. For fast synchronization to UTC, I propose a novel timekeeping software, whose transient components converge after less than a minute. However, the drawback of my software is a lower oscillator stability compared to traditional timekeeping software.

The main findings of this work are listed below. First, modern computers are able to reach a timekeeping precision of roughly 10 μs, but also older models (e.g., DANTE, 2002) show quite good performance. The software clock has a synchronization offset with positive mean (timing lag), with an order of magnitude higher than the respective standard deviation. Hence, compensation for the timing lag yields a clock with nearly microsecond precision. Unfortunately, the measurement of the lag is a complex task.

Second, the interrupt latency is the factor which affects the timing lag of the software clock strongest. This interrupt latency, in turn, depends almost exclusively on the chipset of the PC, whereas CPU type (comparisons between Table C.1 and Figure C.5 (a)) and interface card, cf. Figure C.6 (a–b), have been confirmed to have minor influence.

The measurements performed within this work are based on the temporal difference between an electrical pulse output at the parallel port of the host and the absolute time recorded by the software upon this action. Thereby the delay between (i) the software giving the command for outputting a pulse and (ii) the actual appearance of the pulse at the parallel interface is not known exactly. However, it has been assessed by companion measurements, which show that it is around 1–2 μs, cf. Section C.2. This delay causes a substantial uncertainty in the synchronization accuracy measurements; especially, in cases of very tight synchronization ($< 10\,\mu$s). The remedy is to tap the pulse on a different interface (e.g., directly at the processor). This approach is however not considered, because it would require hardware modifications and involved decoding procedures which may be different on different computer architectures.

Appendix D

Field Trial: Benchmarking Public Mobile Networks

In the following I demonstrate latency measurements in both link directions for various mobile network technologies, namely, Wireless Local Area Network (WLAN), Long Term Evolution (LTE) and High Speed Packet Access (HSPA). The first network type, WLAN, is neither public nor mobile as such, however, it is analyzed for: (i) It is among the most widely deployed wireless technologies. (ii) It exhibits non-reactive behavior, thus, serves as good reference to illustrate reactive effects.

D.1 Measurement Setup

The measurement setup consists of two computers exchanging probing packets: (i) the client, which uses the *modem under test* to connect to the Internet and (ii) the server, connected to the university backbone via Gigabit-Ethernet. The modems deployed for the measurements are, depending on the technology, (i) WLAN: *TP-LINK, TL-WN821N*, (ii) LTE: *HUAWEI, E392* and (iii) HSPA: *NOKIA, CS-18*. They were located in an office environment, strictly at the same position and with line-of-sight connection to the respective base station, for all measurements (i.e., all technologies and providers).
For negotiating the measurement connection setup (ports, NAT-addresses, start time) and exchanging measurement results, a control line is established between both machines. This guarantees that only probing traffic is exchanged over the measurement interfaces. Both PCs run *Linux* and are synchronized to Coordinated Universal Time (UTC) by the *LinuxPPS* driver, which enables a synchronization accuracy of roughly $10\,\mu$s [6]. This accuracy is satisfactory for the measurement purpose, since I aim to measure delays being always bigger than a few hundreds of microseconds. The actual measurement, namely, the capturing of packets and the respective timestamping, is performed by the *libpcap* driver [114] and the *tshark* software [106]; hence, the term latency refers to end-to-end One-Way Delay (OWD) in the kernel domain.
The packets exchanged between both computers are User Datagram Protocol (UDP) datagrams. UDP is popular for real-time services and lacks any flow-control mechanisms which could affect the latency. The corresponding ports are 6000 and 6001, for downlink and uplink, respectively. Due to packet fragmentation and possible duplication there may be multiple timestamps per packet, yielding an ambiguity for the definition of latency; for such cases I define latency as the difference in time between the last fragment captured at the source side and the last fragment captured on the destination side, for reasons outlined in Section 2.2. Synchronization between uplink and downlink direction

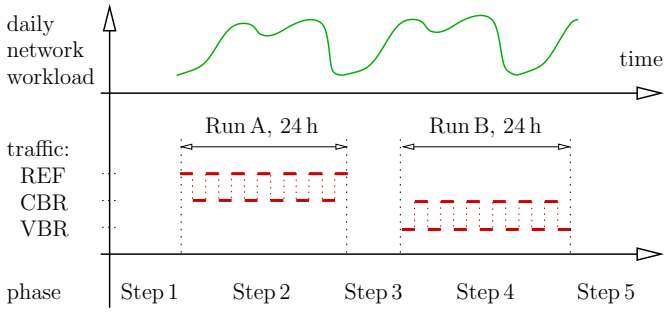


FIGURE D.1: Network benchmarking strategy including two 24 h measurement runs with different interleaved probing patterns.

was suppressed by short-term randomization, which is important according to [129]; the global traffic statistics (patterns), however, where strictly the same in both directions. A valid latency sample is obtained under the following conditions: (i) the packets captured at the source and the destination have the same IP header, (ii) they have the same UDP header, (iii) they have the same first 10 B of payload, (iv) they pass both interfaces within 1.5 s, (v) the clocks of both computers have less than 50 μs deviation from UTC, (vi) the measurement run is finished without error and (vii) less than 5% of all packets within the run have no matching packet at the other interface (packet loss).

D.2 Data Set

In Section D.3 results for multiple networks are presented, including one WLAN network, two LTE networks and three HSPA networks; each performed between October 2012 and April 2013. One such study comprises two 24 h measurement runs, one for Constant Bit-Rate (CBR) traffic introduced in Step 2, the other for Variable Bit-Rate (VBR) traffic according to Step 4 (cf. Section 4.1.4). The total amount of packets exchanged per network is roughly three million, corresponding to 8 GB of data.
Conducting statistically significant measurements with VBR and CBR traffic patterns requires a measurement duration of roughly 2 h and 1 h, respectively. Additionally, measurements deploying the Reference (REF) probing pattern require also 2 h. The reason why I yet perform 24 h measurements for both cases is the potentially strong influence from slowly changing parameters of Φ, especially diurnal patterns (cf. Section 2.3.2, Table 2.3). Since the whole measurement study cannot be performed within T_ϕ (i.e., 5 h compared to 15 min), long-term changes inevitably disturb each study. The remedy is to perform multiple such trials over one day and combine them to a daily average [118]. The stability and reproducibility of this average is tested in Figure 4.3 (b). It compares two CBR measurements, roughly one week apart, which pass the consistency test proposed in Section 4.1.4.5 for most combinations in the Packet Size (PS)-Inter Packet-Arrival Time (IAT) plane.
Measurements with VBR, CBR and REF traffic should be conducted in an interleaved fashion. The problem is thereby that the VBR patterns can only be derived after Step 2 has been completed (after the entire CBR measurement). For this reason, two 24 h measurement series are performed: The first (Run A) is required to determine the cluster structure of the network (Step 2, 3), whereby interleaved measurements are performed with CBR and REF traffic for the validation of the assumptions (Step 5). The second measurement run (Run B) is required to perform interleaved measurements deploying VBR and CBR traffic in order to obtain comparable results for both traffic patterns; or,

equivalently, to suppress any long-term changes from the former day to the present day. A respective graphical representation is given in Figure D.1.

D.3 Results

The comprehensive measurement results are presented in three figures per link direction, confer Figure 4.2, D.2–D.6. The first one reveals the structure of the delay clusters of the network, the second exhibits the respective delay behavior for CBR traffic and the third compares VBR traffic to CBR traffic. This representation is advantageous for (i) ease of interpretation, (ii) the vast amount of contained information and (iii) the clusters serve as bounds for general traffic patterns. Thus, if a traffic pattern is extended over multiple clusters, its latency response is expected to be bounded by the best and worst of the respective Empirical Cumulative Distribution Functions (ECDFs).

D.3.1 WLAN Measurements

WLAN measurements have already been presented in Section 4.1.4 and Figure 4.2, in order to explain the basic concept of the proposed measurement strategy. This is due to respective behavior, being in good accordance with the behavior expected for a non-reactive network. The typical distinguishing marks are: (i) horizontal cluster borders, cf. Figure 4.2 (a, d) (ii) the latency ECDFs for each cluster are shifted versions of each other with touching confidence intervals, cf. Figure 4.2 (b, e) and (iii) negligible differences between the reactions on CBR and VBR traffic, cf. Figure 4.2 (c, f). In such a situation two measurement points (PS-IAT tuples) would be sufficient to determine the delay response of the network for any kind of input traffic.
Comparing these results to modern mobile communication networks, such as LTE and HSPA networks, reveals strong differences. Figure D.2–D.3 show the latency figures for two public LTE networks, whereas Figure D.4–D.6 reveal the behavior of three public HSPA networks. It is observable, that the three distinguishing marks for non-reactive networks are not present in neither of the five cases. Refined measurement methods (as the present one) are undoubtedly required for the delay performance assessment of such networks, especially, when hidden details about the network shall be assessed.

D.3.2 LTE Measurements

Figure D.2–D.3 show the latency responses of the LTE networks of Provider 1 and Provider 2. In general, all measured OWD ECDFs are strongly concentrated, or equivalently, show negligible variance. This is an indicator for a low workload (especially of the core network). High workload would cause delay fluctuations (jitter) and, consequently, a higher variance (cf. [3]). Both networks and both link directions exhibit similarities to non-reactive network behavior (cf. Eq. (2.10)): (i) horizontal cluster borders and (ii) negligible differences in the reactions on CBR and VBR traffic. However, there are some clear indications that the delay response of LTE networks is strongly non-linear. In the uplink direction: (i) The cluster borders are not equally spaced over the PS (in linear scale), indicating that there are thresholds in the PS on which the network reacts. (ii) The delay ECDFs are not shifted replicas of each other, which is confirming the above assumption. (iii) The confidence intervals on the latency ECDFs are not touching each other (e.g., Cluster 1 and Cluster 2). This effect is expected to be caused by pre-allocated resources which allow for lower delay for packets below 100 B. (iv) Figure D.2 (a) shows a toggling between Cluster 3 and Cluster 4 at PSs of 2–5 kB. This effect is accredited

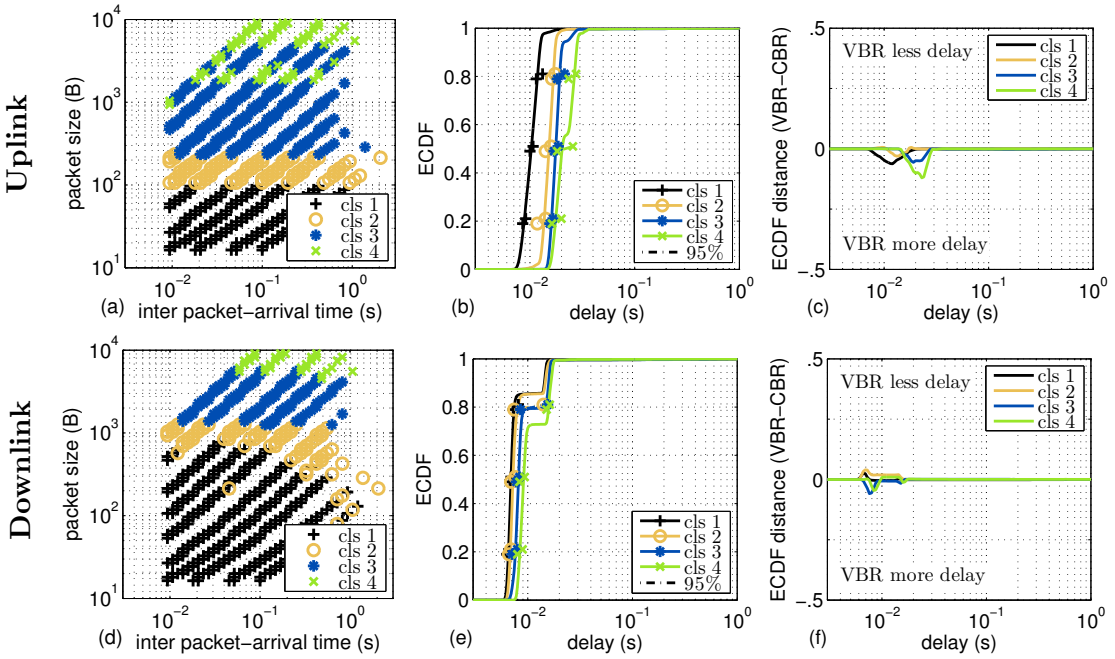


FIGURE D.2: LTE, Provider 1.

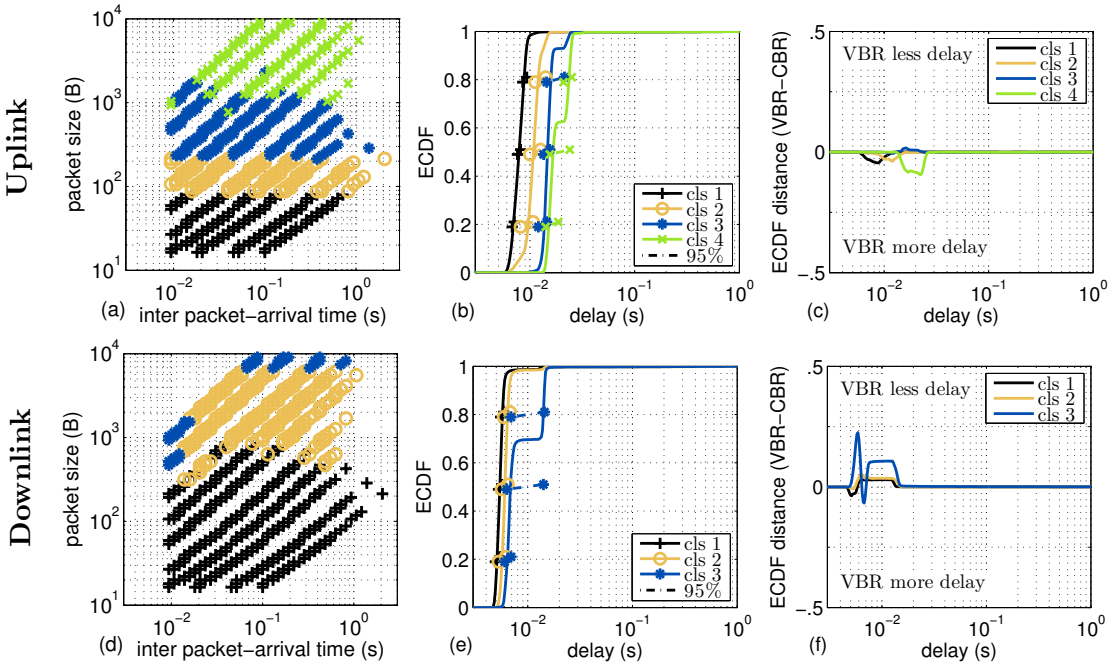


FIGURE D.3: LTE, Provider 2.

to the LTE specific channel coding procedure, which yields blocks of certain sizes less protected than others; hence, more likely to be retransmitted. In the downlink direction non-linear effects manifest in: (i) The latency ECDFs of different cluster all have a plateau, which however changes in hight. This effect is caused by fast retransmissions, which are more frequent for bigger packets than for smaller packets. (ii) Figure D.3 (c) shows a bipartite shape for Cluster 3. Retransmissions seem to occur more frequently within both areas, indicating resource allocation problems. (iii) The above mentioned effect is suppressed by deploying VBR traffic, see Figure D.3 (f); indicated by a latency

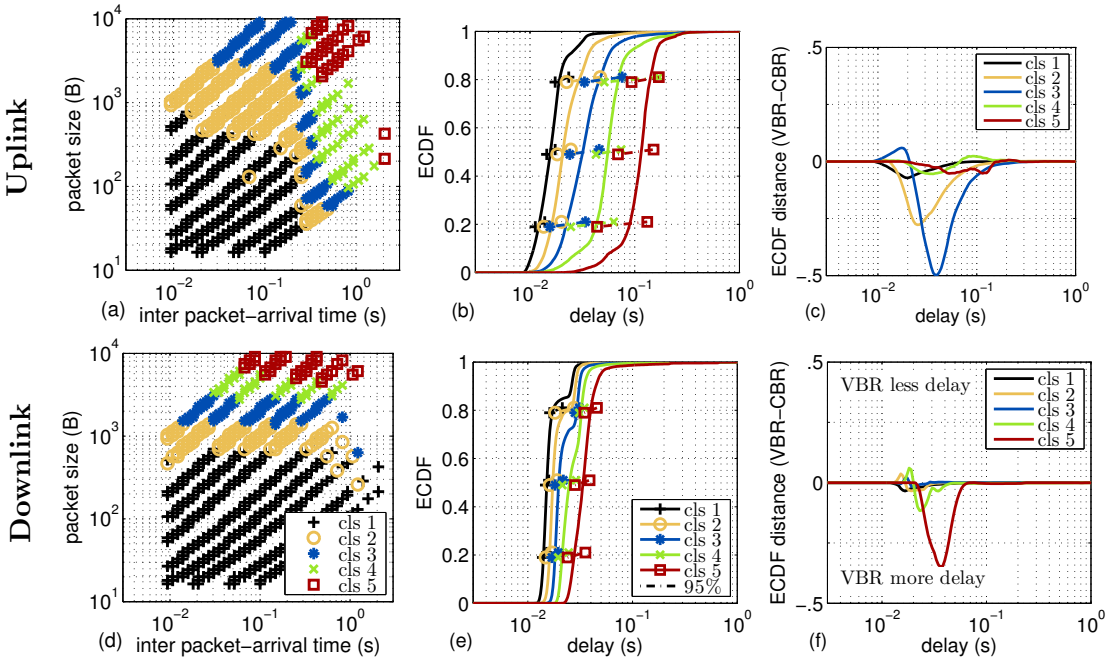


FIGURE D.4: HSPA, Provider 1.

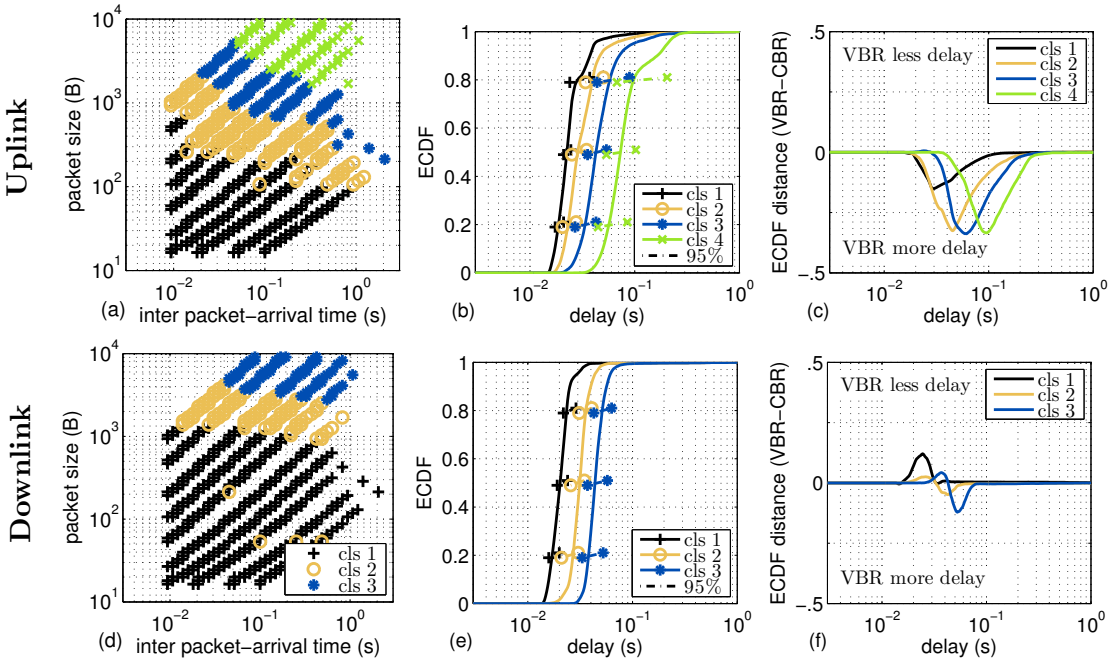


FIGURE D.5: HSPA, Provider 2.

reduction of almost 50% for the higher quantiles, which is equivalent to lift the plateau of the ECDF to around 0.95.

Both networks exhibit a similar structure of clusters in the PS-IAT plane. Also the respective latency ECDFs show resemblance. However the network of the 1st provider is in both link directions 2 ms slower than the network of the second one. Compared to the overall OWD this amounts to a latency increase of roughly 30%. A further difference between the two networks are the retransmission ratios in downlink, identifiable by the plateaus in the latency ECDFs. Whereas the network of Provider 1 shows retransmission

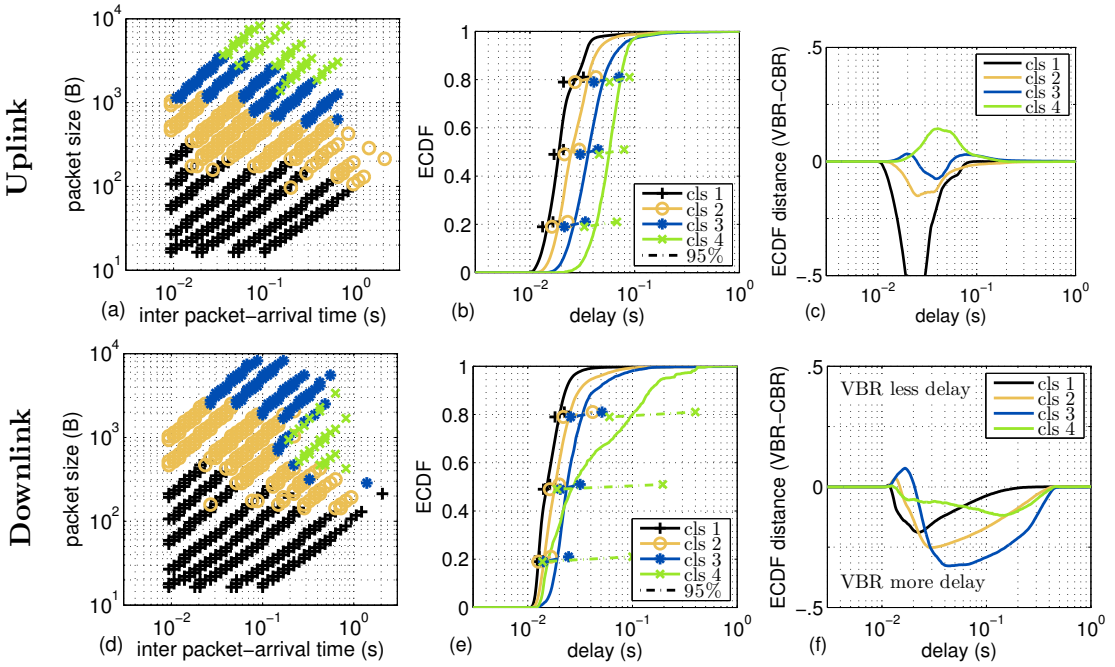


FIGURE D.6: HSPA, Provider 3.

ratios around 10-15% for small packet sizes, the network of Provider 2 shows in general a negligible amount of retransmissions.

D.3.3 HSPA Measurements

The latency responses of three HSPA networks are analyzed in Figure D.4–D.6. The individual latency ECDFs are relatively wide, the 90th quantile may be by a factor of three higher than the 10th quantile, cf. Figure D.4 (b). This indicates strong fluctuations caused by Ψ attributable to a high system workload. Unlike the analyzed LTE networks, HSPA shows prominent reactive behavior: (i) In most cases the cluster boundaries are not horizontal, neither in uplink nor in downlink. (ii) The latency ECDFs for each cluster show different shapes and not only different means. (iii) VBR traffic has a strong influence on the delay figures, e.g., Figure D.6 (f) suggests that VBR traffic may experience a median delay which is 40% higher than the respective median CBR delay. The uplink direction exhibits the more remarkable behavior, due to fundamental differences in resource allocation procedure [22]. Thereby the overall tendency to transmit packets faster (lower delay) when the respective inter-arrival time is lower can be observed; yielding diagonal lines as cluster borders in the PS-IAT plane. Further, the reaction on variations in the instantaneous data rate is distinctive. In general, packets are transmitted faster when they appear within a highly predictable CBR stream.
Comparing the three networks explicitly exposes the relevance of the probing pattern for latency measurements. For example, consider a probing stream with PS of 100 B and IAT of 10 ms in uplink: the network of Provider 1 performs best (Cluster 1, median uplink OWD: 16 ms), followed by Provider 3 (Cluster 1, 19.4 ms) and Provider 2. (Cluster 1, 25.8 ms), confer Table 4.1. Only changing the IAT to 1 s draws a different picture: now the network of Provider 3 performs best (Cluster 1, median uplink OWD: 19.4 ms), followed by Provider 2 (Cluster 2, 32.1 ms) and Provider 1 (Cluster 4, 58.4 ms). This yields a fair ranking among multiple networks only possible if refined probing strategies are deployed. Further, the trade-off between optimizing the network settings for a certain

traffic type and accepting disadvantages for other kinds of traffic becomes visible by comparing different networks (e.g., uplink, Provider 1: optimized network for low IAT; Provider 2: no obvious optimization).
The measurement methodology proposed in Section 4.1.4 allows for an extensive evaluation of networks. Undesired but hidden effects can be detected with reasonable accuracy and effort. For example, the network of Provider 3 shows an interesting artifact in downlink, namely, Cluster 4, cf. Figure D.6 (e, f), which is restricted to a rather small area in the PS-IAT plane. The respective delay response would be expected to resemble that of Cluster 2 or Cluster 3 (compare with other providers), however, the latency ECDF shows a strong tail and the respective confidence intervals are roughly comprising a whole decade (20 ms–200 ms). This effect, although not affecting the most prominent traffic patterns, indicates some room for improvements at the network side.

Appendix E

Derivation of Analytical Expressions for Transformed Gaussian ARMA Models

Here I present analytical results for the properties of the Gaussian Transformed Auto-Regressive Moving-Average (TARMA) processes $Z_i[n]$, presented in Section 5.1. The relations concerning the polynomial transformation, namely, Eq. (5.3), Eq. (5.4) and Eq. (5.5), are rarely concerned in literature [186, pp. 419–426] and therefore established in the following. The expressions are usually provided in terms of Hermite polynomials which can be found in [189, pp. 132ff.] [185] [188].
For convenience the following functions are defined:
• the binomial coefficient extended to all integer numbers

$$\binom{l}{k} \doteq \begin{cases} \frac{l!}{k!(l-k)!} & \text{if } 0 \leq k \leq l, \\ 0 & \text{otherwise,} \end{cases} . \tag{E.1}$$

• the multinomial coefficient

$$\binom{l}{k_1, k_2, \cdots, k_P} \doteq \frac{l!}{k_1! \cdot k_2! \cdots k_P!}, \tag{E.2}$$

• the double factorial operator, extend to negative values,

$$(k)!! \doteq \begin{cases} 1 \cdot 3 \cdots (k-2) \cdot k & \text{if } k > 0 \text{ and odd,} \\ 2 \cdot 4 \cdots (k-2) \cdot k & \text{if } k > 0 \text{ and even, ,} \\ 1 & \text{if } k \leq 0, \end{cases} \tag{E.3}$$

• the indicator function for parity

$$\mathbf{1}_e(k) \doteq \begin{cases} 0 & \text{if } k \text{ odd,} \\ 1 & \text{if } k \text{ even.} \end{cases} \tag{E.4}$$

E.1 Distribution

The polynomial $\mathfrak{p}_Y(y)$ introduced in Eq. (5.1) is fit to the quantile transformation function, $\mathfrak{p}_Y(y) \approx F^{-1}_{Z,\text{target}}(F_Y(y))$. Thereby, the goal is to achieve $f_Z(y) \approx f_{Z,\text{target}}(y)$ where $f_{Z,\text{target}}(y)$ is the targeted Probability Density Function (PDF), corresponding to the

inverse Cumulative Distribution Function (CDF) $F^{-1}_{Z,\text{target}}(\cdot)$, and $f_Z(y)$ is the actually achieved PDF of the output process of the model $Z[n]$, see Corollary E.1. Polynomials with high order P are guaranteed to match at least P points of any arbitrary function [195]; hence, any targeted PDF can be reproduced with arbitrary accuracy by the modeling approach. Yet, since low-order polynomials are especially attractive for any low-complexity implementation of the model, the discrepancy between targeted PDF and actually achieved PDF is of interest, in order to assess the quality of the fit.
In order to determine the PDF of the random process $Z[n]$, the theorem on transformation of random variables is consulted, cf. Theorem E.2, out of which the following corollary is extracted:

Corollary E.1

Consider a random process

$$Z[n] = \mathfrak{p}_Y(Y[n]) = \sum_{p=0}^{P} \alpha_p \cdot (Y[n])^p,$$

where $Y[n]$ denotes a stationary Gaussian random process with zero mean and unit variance. To find the PDF $f_Z(z)$ of $Z[n]$ for a specific value z, the equation $\mathfrak{p}_Y(y) - z = 0$ must be solved. Denoting its real roots by y_k, with cardinality K and $z = \mathfrak{p}_Y(y_1) = \cdots = \mathfrak{p}_Y(y_K)$, the density conforms to

$$f_Z(z) = \begin{cases} \displaystyle\sum_{k=1}^{K} \frac{\exp(-y_k^2/2)}{\sqrt{2\pi}\,|\sum_{p=1}^{P} p \cdot a_p \cdot (y_k)^{p-1}|} & \text{if } K > 0, \\ 0 & \text{otherwise,} \end{cases}$$

where $|\cdot|$ denotes the absolute value.

For the derivation of this corollary I consult the fundamental theorem on transformations of random variables (cf. [164, p. 130]):

Theorem E.2 (*Fundamental Theorem on Transformation of Random Variables*)

To find the PDF $f_Z(z)$ of a random process $Z[n] = \mathfrak{p}_Y(Y[n])$ for a specific z, the equation $z = \mathfrak{p}_Y(y)$ has to be solved. Denoting its real roots by y_k with

$$z = \mathfrak{p}_Y(y_1) = \cdots = \mathfrak{p}_Y(y_k) = \cdots = \mathfrak{p}_Y(y_K),$$

it becomes apparent that

$$f_Z(z) = \frac{f_Y(y_1)}{|\mathfrak{p}'_Y(y_1)|} + \cdots + \frac{f_Y(y_k)}{|\mathfrak{p}'_Y(y_k)|} + \cdots,$$

where $\mathfrak{p}'_Y(\cdot)$ denotes the derivative of $\mathfrak{p}_Y(\cdot)$ and $|\cdot|$ the absolute value.

Further, plugging the Gaussian PDF and the derivative of a polynomial into this theorem immediately yields Corollary E.1.

E.2 Moments

In the following the moments $\mathsf{m}_Z^{(l)} = E\{(Z[n])^l\}$ of the marginal distribution of $Z[n]$ are derived, with $E\{\cdot\}$ denoting the expectation operator. Note that the extension

to mixed moments is possible (e.g., Auto-correlation Function (ACF) in Section E.3, Cross-correlation Function (XCF) in Section E.4), however, neglected in this section for simplicity. The derivation of the moments of the marginal of $Z[n]$, as well as the derivations of ACF of $Z[n]$ and XCF of the processes $Z_i[n]$ is based on Isserlis' theorem [248], see Theorem E.4. The theorem allows to calculate any moment of multivariate Gaussian distributions. This is convenient for our case, since the marginal moments $\mathsf{m}_Z^{(l)}$ of the process $Z[n]$ arise from the polynomial transformation of the Gaussian process $Y[n]$, hence, are linear combinations of higher order marginal moments $\mathsf{m}_Y^{(l)}$ of the Gaussian process $Y[n]$. The following expression for the marginal moments of $Z[n]$ is obtained, formulated in terms of random variables for generality:

Corollary E.3

If a Normal random variable Y with zero mean and unit variance is transformed by a polynomial $\mathfrak{p}_Y(\cdot)$ to a random variable $Z = \mathfrak{p}_Y(Y) = \sum_{p=0}^{P} \alpha_p \cdot (Y)^p$, then the moments $\mathsf{m}_Z^{(l)}$ of the variable Z equal

$$\mathsf{m}_Z^{(l)} = \sum_{k_0+k_1+\cdots+k_P=l} \binom{l}{k_0, k_1, \cdots, k_P} \cdot \left(\prod_{p=0}^{P} \alpha_p^{k_p} \right) \cdot \left(\sum_{p=0}^{P} p \cdot k_p - 1 \right)!! \cdot \mathbf{1}_e\left(\sum_{p=0}^{P} p \cdot k_p \right),$$

where l denotes the order of the moment, $\binom{l}{k_0,k_1}$ denotes the multinomial coefficient, Eq. (E.2), $(\cdot)!!$ is the double factorial coefficient, Eq. (E.3), and $\mathbf{1}_e(\cdot)$ the indicator function for parity, Eq. (E.4). Thereby the summation is performed over all sequences of nonnegative integers k_0 through k_P such that the sum of all k_p equals l. The number of terms in the sum equals $\binom{l+P}{l}$.

The derivation of this corollary is provided below. Further consider two special cases of this corollary, utilized in the following sections; namely, the mean and the variance. Accordingly, the mean of the process $Z[n]$ calculates to

$$\mu_Z = \mathsf{m}_Z^{(1)} = \sum_{p=0}^{P} \alpha_p \cdot (p-1)!! \cdot \mathbf{1}_e(p). \tag{E.5}$$

The variance of $Z[n]$, after collapsing the coefficients of the summation to a squared form, calculates to

$$\sigma_Z^2 = \mathsf{m}_Z^{(2)} - \mu_Z^2 = \sum_{k=1}^{P} k! \left(\sum_{p=0}^{P} \binom{p}{k} \alpha_p \, (p-k-1)!! \, \mathbf{1}_e(p-k) \right)^2. \tag{E.6}$$

The basic theorem deployed for the derivation of this corollary is Isserlis' theorem [248] and, moreover, the multinomial theorem [249].

Theorem E.4 (*Isserlis' Theorem*)

If $(x_1, x_2, \cdots, x_{2n})$ is a zero mean multivariate normal vector, then

$$E\{x_1 x_2 \cdots x_{2n}\} = \sum \prod E\{x_i x_j\},$$
$$E\{x_1 x_2 \cdots x_{2n-1}\} = 0,$$

where $E\{\cdot\}$ denotes the expectation-operation and the notation $\sum\prod$ means summing over all distinct ways of partitioning $x_1, x_2, \cdots, x_{2n}$ into pairs.

Theorem E.5 (*Multinomial Theorem*)

For any positive integer P and any nonnegative integer l a polynomial expands according to

$$(x_0 + x_1 + \cdots + x_P)^l = \sum_{k_0+\cdots+k_P=l} \binom{l}{k_0, \cdots, k_P} \prod_{p=0}^{P} x_p^{k_p},$$

where the sum is taken over all sequences of nonnegative integer indexes k_p such that they sum to l and $\binom{l}{k_0,\cdots,k_P}$ denoting the multinomial coefficients, cf. Eq. (E.2). There are $\binom{l+P}{l}$ terms in the multinomial sum.

If the random variable Z is a polynomial transformation of Y, cf. Eq. (5.1), then its l-th moment can be determined to

$$\begin{aligned}
\mathsf{m}_Z^{(l)} &= E\{Z^l\} = E\{\mathfrak{p}_Y^l(Y)\} \\
&= E\Big\{ \big(\sum_{p=0}^{P} a_p \cdot Y^p\big)^l \Big\} \\
&= E\Big\{ \sum_{k_0+\cdots+k_P=l} \binom{l}{k_0, \cdots, k_P} \prod_{p=0}^{P} (a_p Y^p)^{k_p} \Big\} \\
&= \sum_{k_0+\cdots+k_P=l} \binom{l}{k_0, \cdots, k_P} \big(\prod_{p=0}^{P} a_p^{k_p}\big) E\big\{Y^{\sum_{p=0}^{P} p\cdot k_p}\big\}.
\end{aligned}$$

Using Isserlis' Theorem for zero mean multivariate normal random variables Y (cf. Theorem E.4) and $\zeta = \sum_{p=0}^{P} p \cdot k_p$ the term $E\{Y^\zeta\}$ further modifies to

$$E\{Y^\zeta\} = (\zeta - 1)!! \cdot \mathbf{1}_e(\zeta) \cdot E\{Y^2\}^{\zeta/2},$$

where $(\cdot)!!$ is the double factorial operation, see Eq. (E.3), and $\mathbf{1}_e(\cdot)$ denotes the parity indicator, defined in Eq. (E.4). Plugging this expression back into $m_Z^{(l)}$ and accounting for $E\{Y^2\}^{\zeta/2} = \sigma_Y^\zeta = 1^\zeta = 1$, due to the definition of Y from Eq. (5.7), Corollary E.3 is directly obtained.

E.3 Auto-correlation Function

For the derivation of the auto-correlation function of the random process $Z[n]$, the following theorem is established first:

Theorem E.6

If Y_1 and Y_2 are Gaussian random variables with zero mean and unit variance, which are transformed by two polynomials into the random variables Z_1 and Z_2

by

$$Z_1 = \mathfrak{p}_{Y,1}(Y_1) = \sum_{p=0}^{P} \alpha_p \cdot (Y_1)^p,$$

$$Z_2 = \mathfrak{p}_{Y,2}(Y_2) = \sum_{q=0}^{Q} \beta_q \cdot (Y_2)^q,$$

and the unnormalized correlation of Y_1 and Y_2 is denoted $E\{Y_1Y_2\}$, then the correlation of Z_1 and Z_2 equals

$$E\{Z_1Z_2\} = \sum_{k=0}^{\min(P,Q)} k! \cdot \big(E\{Y_1Y_2\}\big)^k \cdot \left(\sum_{p=0}^{P} \alpha_p \cdot \binom{p}{k} \cdot (p-k-1)!! \cdot \mathbf{1}_e(p-k)\right) \cdot \left(\sum_{q=0}^{Q} \beta_q \cdot \binom{q}{k} \cdot (q-k-1)!! \cdot \mathbf{1}_e(q-k)\right).$$

To prove this theorem, the following lemma is constituted

Lemma E.7

If Y_1 and Y_2 are two Gaussian random variables, then the expectation of the product of an arbitrary power of them $E\{Y_1^pY_2^q\}$ is determined to

$$E\{Y_1^pY_2^q\} = \mathbf{1}_e(p+q) \cdot \sum_{k=0}^{\min(p,q)} \mathbf{1}_e(q-k) \cdot k! \cdot E\{Y_1Y_2\}^k \cdot \left(\binom{p}{k} \cdot (p-k-1)!! \cdot E\{Y_1Y_1\}^{(p-k)/2}\right) \cdot \left(\binom{q}{k} \cdot (q-k-1)!! \cdot E\{Y_2Y_2\}^{(q-k)/2}\right),$$

with $\mathbf{1}_e(\cdot)$ denoting the indicator function for parity, cf. Eq. (E.4), $(\cdot)!!$ the double factorial operator, see Eq. (E.3), and, $\binom{l}{k}$ the binomial coefficient, cf. Eq. (E.1).

Proof: According to Isserlis' Theorem (cf. Theorem E.4) it is possible to calculate this moment of a multivariate Gaussian random variable by summation over all distinct ways of partitioning the p+q random variables $(Y_1, \cdots, Y_1,\ Y_2, \cdots, Y_2)$ into pairs. Let the set of the p variables Y_1 be denoted $\mathbb{P}$ and the set of q variables Y_2 be denoted $\mathbb{Q}$. Further, $\mathbb{P}$ is fragmented into two sets, $\mathbb{V}_1$, with k variables Y_1, and $\mathbb{W}_1$, containing $p-k$ variables Y_1. Also $\mathbb{Q}$ is partitioned into two sets, namely, $\mathbb{V}_2$, with k variables Y_2 and $\mathbb{W}_2$, with $q-k$ variables Y_2. Combining variables from $\mathbb{W}_1$ and $\mathbb{W}_2$ internally, but those from $\mathbb{V}_1$ with $\mathbb{V}_2$, yields one possible combination of pairs, hence, the coefficient

$$E\{Y_1Y_2\}^k \cdot E\{Y_1^2\}^{(p-k)/2} \cdot E\{Y_2^2\}^{(q-k)/2}$$

is one coefficient in the sum of Isserlis' Theorem. Thereby, $p-k$ must be an even number ($\mathbf{1}_e(p-k)$), which guarantees that also $q-k$ is an even number, because $p+q$ is even by definition of the theorem.

There are $(p-k-1)!!$ distinct way to form pairs of variables from the set $\mathbb{W}_1$ [250] and $(p-k-1)!!$ ways to form pairs of variables from $\mathbb{W}_2$. Furthermore, there are $k!$ different ways to from pairs by combining one variable from $\mathbb{V}_1$ with one from $\mathbb{V}_2$. All these possible combinations yield the same coefficient, shown in the equation above. Moreover, there are $\binom{p}{k}$ different ways to choose $\mathbb{V}_1$ from $\mathbb{P}$ and $\binom{q}{k}$ different ways to choose $\mathbb{V}_2$ from $\mathbb{Q}$, all of which again yield the same coefficient in the sum of Isserlis' Theorem.

Now, collapsing all coefficients with the same cardinality $|\mathbb{V}_1| = |\mathbb{V}_2| = k$ into a product and summing over all possible cardinalities k, which range from zero to the minimum of P and Q, yields Lemma E.7. ■

Consequently, Theorem E.6 can be proved by introducing two new random variables Z_1 and Z_2, descending from polynomial transformations of the Gaussian random variables Y_1 and Y_2,

$$Z_1 = \mathfrak{p}_{Y,1}(Y_1) = \sum_{p=0}^{P} a_p \cdot (Y_1)^p,$$

$$Z_2 = \mathfrak{p}_{Y,2}(Y_2) = \sum_{q=0}^{Q} b_q \cdot (Y_2)^q.$$

Proof: The expectation of the product of both random variables equals

$$\begin{aligned}
E\{Z_1 Z_2\} &= E\{\mathfrak{p}_{Y,1}(Y_1) \cdot \mathfrak{p}_{Y,2}(Y_2)\} \\
&= E\Big\{ \big(\sum_{p=0}^{P} a_p \cdot (Y_1)^p \big) \cdot \big(\sum_{q=0}^{Q} b_q \cdot (Y_2)^q \big) \Big\} \\
&= \sum_{p=0}^{P} \sum_{q=0}^{Q} a_p b_q \cdot E\{Y_1^p Y_2^q\} \\
&= \sum_{p=0}^{P} \sum_{q=0}^{Q} a_p b_q \cdot \\
&\Bigg(\mathbf{1}_e(p+q) \cdot \sum_{k=0}^{\min(p,q)} \mathbf{1}_e(q-k) \cdot k! \cdot E\{Y_1 Y_2\}^k \cdot \\
&\left(\binom{p}{k} \cdot (p-k-1)!! \cdot E\{Y_1 Y_1\}^{(p-k)/2} \right) \cdot \\
&\left(\binom{q}{k} \cdot (q-k-1)!! \cdot E\{Y_2 Y_2\}^{(q-k)/2} \right) \Bigg),
\end{aligned}$$

where Lemma E.7 is used for the last manipulation. Inserting the identity $\mathbf{1}_e(p+q) \cdot \mathbf{1}_e(p-k) = \mathbf{1}_e(p-k) \cdot \mathbf{1}_e(q-k)$ and the definition $E\{Y_i Y_i\} = 1$ and interchanging

the summation order results in

$$E\{Z_1 Z_2\} = \sum_{k=0}^{\min(P,Q)} k! \cdot E\{Y_1 Y_2\}^k \cdot \left(\sum_{p=k}^{P} a_p \binom{p}{k} \cdot (p-k-1)!! \cdot \mathbf{1}_e(p-k) \right) \cdot \left(\sum_{q=k}^{Q} b_q \binom{q}{k} \cdot (q-k-1)!! \cdot \mathbf{1}_e(q-k) \right),$$

Finally, because of the definition of the binomial coefficient in Eq. (E.1), which is zero for l smaller k, the summation can be extended over p and q from zero to P and Q, respectively, what proves Theorem E.6. ■

The auto-correlation function $\rho_{ZZ}[m]$ of the process $Z[n]$ can now be determined by plugging the statement of this theorem into Eq. (5.8), together with Eq. (E.5), which changes the summation limit from $k = 0$ to $k = 1$, and Eq. (E.6), which causes a scaling of the coefficients. The following corollary is obtained, equivalent to Eq. (5.4).

Corollary E.8

Let $Y[n]$ denote a Gaussian random process with zero mean, unit variance and auto–correlation function $\rho_{YY}[m]$ and $Z[n]$ the random process obtained by the transformation of $Y[n]$ by a polynomial $\mathfrak{p}_Y(\cdot)$ according to $Z[n] = \mathfrak{p}_Y(Y[n]) = \sum_{p=0}^{P} \alpha_p \cdot (Y[n])^p$. Then the auto–correlation function of the random process $Z[n]$ equals

$$\rho_{ZZ}[m] = \mathfrak{p}_\rho(\rho_{YY}[m]) = \sum_{k=1}^{P} \xi_k \cdot \big(\rho_{YY}[m]\big)^k,$$

where $\mathfrak{p}_\rho(\cdot)$ denotes a polynomial with coefficients ξ_k, which, for $k = 1, \cdots, P$ are calculated to

$$\xi_k = \frac{1}{\sigma_Z^2} k! \left(\sum_{p=0}^{P} \alpha_p \cdot \binom{p}{k} \cdot (p-k-1)!! \cdot \mathbf{1}_e(p-k) \right)^2,$$

where σ_Z^2 denotes the variance of $Z[n]$, see Eq. (E.6)).

This results specifies the ACF of the output process $Z[n]$ in closed form, being crucial for an efficient fitting procedure, as described in Section 5.1.3. Note that the order of the polynomial $\mathfrak{p}_\rho(\cdot)$ is equal to the order P of the polynomial $\mathfrak{p}_Y(\cdot)$. Furthermore, the calculation of a single coefficient α_k requires the summation over P terms only. Hence, the computational complexity for the determination of $\mathfrak{p}_\rho(\cdot)$ is very low.

E.4 Cross-correlation Function

The cross-correlation $\rho_{Z_1 Z_2}[m]$ of two random processes $Z_1[n]$ and $Z_2[n]$ can be determined similar as the ACF in Corollary E.8, namely, by plugging Theorem E.6 into Eq. (5.8) and using Eq. (E.5) and Eq. (E.6) for normalization. The result are summarized in the following corollary, equivalent to Eq. (5.5)).

Corollary E.9

Let $Y_1[n]$ and $Y_2[n]$ denote two stationary Gaussian random processes with zero

mean and unit variance, and $\rho_{Y_1Y_2}[m]$ *the cross-correlation function between them. Both processes are transformed by polynomials into the random processes* $Z_1[n]$ *and* $Z_2[n]$*, according to* $Z_1[n] = \mathfrak{p}_{Y,1}(Y_1[n]) = \sum_{p=0}^{P} \alpha_p \cdot (Y_1[n])^p$ *and* $Z_2[n] = \mathfrak{p}_{Y,2}(Y_2[n]) = \sum_{q=0}^{Q} \beta_q \cdot (Y_2[n])^q$*, then the cross-correlation function of the processes* $Z_1[n]$ *and* $Z_2[n]$ *equals*

$$\rho_{Z_1Z_2}[m] = \mathfrak{p}_{\rho,12}(\rho_{Y_1Y_2}[m]) = \sum_{k=1}^{\min(P,Q)} \chi_k \cdot \big(\rho_{Y_1Y_2}[m]\big)^k,$$

where $\mathfrak{p}_{\rho,12}(\cdot)$ *denotes a polynomial with coefficients* χ_k *which, for* $k = 1, \cdots, \min(P, Q)$ *are calculated to*

$$\chi_k = \frac{1}{\sigma_{Z_1}} \left(\sum_{p=0}^{P} \alpha_p \cdot \binom{p}{k} \cdot (p-k-1)!! \cdot \mathbf{1}_e(p-k) \right) \cdot \frac{1}{\sigma_{Z_2}} \left(\sum_{q=0}^{Q} \beta_q \cdot \binom{q}{k} \cdot (q-k-1)!! \cdot \mathbf{1}_e(q-k) \right) \cdot k!.$$

Here σ_{Z_1} *and* σ_{Z_2} *denote the standard deviation of* $Z_1[n]$ *and* $Z_2[n]$*, respectively, see Eq. (E.6).*

Appendix F

Fitting ARMA Models to LRD Time Series

This appendix shows how to fit regression models to Long Range Dependence (LRD) time series. The LRD property of network traffic is influencing the queueing performance [251] and, consequently, the experienced delay figures. Therefore, I consider it as important to deploy traffic models which are able to accurately reproduce the corresponding phenomena. Here I treat the problem to fit ordinary Auto-Regressive Moving-Average (ARMA) models to the mentioned time series, for which a novel fitting algorithm is presented.

As mentioned in Section 5.1.4.1, ARMA models are not able to exactly reproduce LRD time series. This is due to the fact that LRDs imply $\sum_{m=1}^{\infty} |\rho_{ZZ}[m]|=\infty$ [201], which cannot be achieved by ARMA or Markovian modeling approaches. However, respective **approximations** to LRD time series can be designed for any desired sample length N with high quality. Several examples can be found in literature, namely, [252] [180] [189].

The Transformed Auto-Regressive Moving-Average (TARMA) modeling approach, presented in Section 5.1, requires that LRDs are reproduced by an ARMA model; where good accordance at large lags is required. This issue is rarely addressed in literature.

The modeling approach presented below provides a methodology to fit ARMA models to LRD time series with high accuracy. It relies on the the targeted Auto-correlation Function (ACF) as input parameter. This is an advantage for TARMA models, since it is easier to estimate the targeted ACF than the targeted time series (cf. Section 5.1.3); which would be required by Maximum Likelihood (ML) methods. The obtained ARMA model is parsimonious, the model order is directly determined during the fitting procedure. Thereby it is possible to trade off fitting error against model order.

F.1 Considerations on ACFs of ARMA Models

The ARMA recursion outlined in Eq. (5.10), often also written in the form $Y[n]=\frac{\theta(B)}{\phi(B)}X[n]$, can be decomposed by a partial fraction expansion into the form

$$Y[n] = X[n] \cdot \left(\sum_{k=1}^{K} \frac{w_k}{1 - a_k B} \right). \tag{F.1}$$

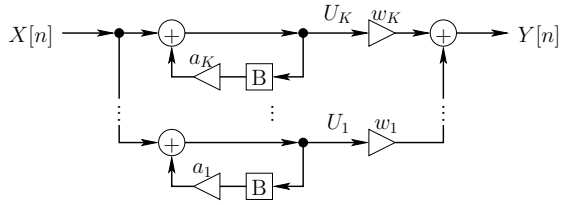


FIGURE F.1: Representation of an ARMA model as sum of exponentials, k=1, ..., K.

A respective graphical representation is given in Figure F.1. Each of the K addends is thereby corresponding to one branch in the figure. Defining the intermediate processes

$$U_k[n] = a_k \cdot U_k[n-1] + X[n] = \sum_{m=0}^{n} a_k^{n-m} \cdot X[m], \tag{F.2}$$

the output process calculates to $Y[n] = \sum_{k=1}^{K} w_k U_k[n]$.
Without loss of generality, I assume the Gaussian input and output processes $X[n]$ and $Y[n]$ to have zero mean and unit variance. Any other distribution can be achieved by a polynomial transformation of the output process as described in Section 5.1. Additionally, $X[n]$ is assumed to be a random uncorrelated (white) sequence, namely, $\rho_{XX}[m]$=$\delta[m]$, where $\delta[m]$ denotes the unit impulse function, yielding $\rho_{XX}[m]$=0 $\forall m \neq 0$. This restricts the values a_k and w_k to

$$|a_k| < 1, \tag{F.3a}$$

$$\sum_{k=1}^{K} \sum_{l=1}^{K} \frac{w_k w_l}{1 - a_k a_l} = 1, \tag{F.3b}$$

for an explanation of Eq. (F.3b), see Eq. (F.7). Accordingly, the ACF of $Y[n]$ is

$$\begin{aligned}
\rho_{YY}[m] &= \frac{\gamma_{YY}[m] - \mu_Y^2}{\sigma_Y^2} = E\left\{Y[n]Y[n+m]\right\} \\
&= E\left\{\left(\sum_{k=1}^{K} w_k U_k[n]\right) \cdot \left(\sum_{l=1}^{K} w_l U_l[n+m]\right)\right\} \\
&= \sum_{k=1}^{K} \sum_{l=1}^{K} w_k w_l \cdot E\left\{U_k[n] U_l[n+m]\right\} \\
&= \sum_{k=1}^{K} \sum_{l=1}^{K} w_k w_l \cdot \gamma_{U_k U_l}[m].
\end{aligned} \tag{F.4}$$

The unnormalized cross-correlation functions of the sequences $U_k[n]$ can further be calculated to

$$\begin{aligned} E\left\{U_k[n]U_l[n+m]\right\} &= E\left\{\left(\sum_{p=0}^{n} a_k^{n-p} X[p]\right)\cdot\left(\sum_{q=0}^{n+m} a_l^{n+m-q} X[q]\right)\right\} \\ &= \sum_{p=0}^{n}\sum_{q=0}^{n+m} a_k^{n-p} a_l^{n+m-q}\cdot E\left\{X[p]X[q]\right\} \\ &= \sum_{p=0}^{n} a_k^{n-p} a_l^{n+m-p} = a_l^m \sum_{p=0}^{n} (a_k a_l)^{n-p} \end{aligned} \tag{F.5}$$

$$\begin{aligned} \gamma_{U_k U_l}[m] &= \lim_{n\to\infty} E\left\{U_k[n]U_l[n+m]\right\} \\ &= \lim_{n\to\infty} a_l^m \sum_{p=0}^{n} (a_k a_l)^{n-p} = \frac{a_l^m}{1-a_k a_l}. \end{aligned} \tag{F.6}$$

Thereby, $\lim_{n\to\infty}$ can be applied in Eq. (F.6) since all involved sequences are stationary and ergodic. Since a_l^m can be written as $a_l^m = \exp(m\cdot\log_e(a_l))$, Eq. (F.4) can be rephrased to

$$\rho_{YY}[m] = \sum_{k=1}^{K}\left(\sum_{l=1}^{K}\frac{w_k w_l}{1-a_k a_l}\right)\cdot\exp\Big(\log_e(a_k)\cdot m\Big). \tag{F.7}$$

Assume that the targeted ACF is fitted by a sum of exponential functions (Section F.2 provides an overview on how to achieve this); then the targeted function can be written as

$$\rho_{YY,\text{target}}[m] \approx \sum_{k=1}^{K}\left(\omega_k\right)\cdot\exp\Big(\alpha_k\cdot m\Big). \tag{F.8}$$

The ARMA coefficients a_k and w_k can be assessed by comparing Eq. (F.7) and Eq. (F.8). This yields the following system of equations:

$$\alpha_k = \log_e(a_k) \qquad \text{and} \qquad \omega_k = \sum_{l=1}^{K}\frac{w_k w_l}{1-a_k a_l}. \tag{F.9}$$

While the first expression can readily be solved for a_k, the second expression is a system of multivariate quadratic equations in w_k. It can be solved by iterative optimization methods [253]; for example, the gradient algorithm, Newton's method or any more advanced approach. Those are standard methods, implemented in many popular software packages. Having obtained a_k and w_k, the partial fraction expansions in Eq. (F.1) are easily collapsed to a rational polynomial in B; the coefficients thereof are the ARMA coefficients under investigation.

F.2 Fitting ACFs by a Sum of Exponentials

In order to determine the ARMA coefficients $\theta(B)$ and $\phi(B)$ according to the above section, it is required to fit a sum of exponential functions to the targeted ACF, as

outlined in Eq. (F.8). Translating the restrictions of Eq. (F.3) to α_k and ω_k yields

$$\mathfrak{Re}(\alpha_k) < 0, \tag{F.10a}$$

$$\sum_{k=1}^{K} \omega_k = 1, \tag{F.10b}$$

where $\mathfrak{Re}(\cdot)$ denotes the real part of a value. Further notice, that in order to obtain a valid ACF for real-valued signals $X[n]$ and $Y[n]$ and with real-valued ARMA coefficients $\theta(B)$ and $\phi(B)$, α_k is allowed to take on complex values, which however must occur in conjugate complex pairs with the same weight ω_k. The values of ω_k are restricted to real positive numbers; otherwise, complex signals would be required to cause the respective ACF.

In the following I present a method for fitting a targeted ACF by complex exponentials. Thereby, the focus is on the accurate modeling of the LRD effects, since they have a strong influence on the queue-length of networks components [254] (especially on the maximum queue-length). This is in contrast to classical methods (e.g., ML estimation, Yule-Walker equations), which aim to determine the ARMA coefficients of the best predictor. For a predictor the accurate modeling of the ACF at low lags is much more important than LRD effects. Consequently, those methods show poor performance at high lags; especially when (i) requirements on parsimoniousness constrains the model order, (ii) the sample size is low compared to the LRD effects or (iii) the samples are polluted by noise or outliers [255] (e.g., in the case of measurements). Therefore, the proposed fitting procedure is better suited to network traffic modeling than classical methods.

Approximating functions by sum of exponentials, as required in Eq. (F.8), is commonly solved by Prony's method or respective enhancements [256]. In the context of data traffic modeling respective approaches have been deployed by [257] [153] [258], in the context of Markovian Arrival Process (MAP) modeling. Prony's method relies on the solution of a linear system of differential equations, where the number of equations is equal to the number of input samples (i.e., maximum lag of the ACF). In order to model LRD a huge number of equations has to be solved, being computationally expensive. This further yields the same number of exponential functions, which have to be collapsed into only few representatives for a parsimonious representation.

Therefore, the proposed approach relies on a different algorithm, which extends the method presented in [46]. In [46] the authors propose to approximate a function recursively by fitting only one exponential to the function at each iteration; where each iteration concerns a different timescale. This procedure matches the LRD nature of network traffic. Although the algorithm has been proposed for approximating Cumulative Distribution Functions (CDFs) by hyper-exponential distributions (cf. [259]), it has been successfully deployed for modeling ACFs by MAP processes [252]. I propose the following algorithm as enhancement to the proposal of [46].

Algorithm F.1 (*Recursive fitting of exponentials to an ACF*)

1. *Choose a maximum lag M_0 (cut-off lag) up to which the ACF shall be modeled. It may be determined by (i) the length of the input time series from which the ACF is estimated or (ii) the longest timescale which is considered to influence the system (cf. [260]).*

2. *Initialize k by k=1 and the intermediate function $\rho_{\text{int},k}[m]$ by the original ACF $\rho_{\text{int},1}[m]$=$\rho_{YY}[m]$.*

3. *Find an appropriate lag M_k, with $M_k<M_{k-1}$, constituting the lower bound of the fitting interval. Details are outlined in Section F.2.1.*

4. *Determine the exponential function $\omega_k \exp(\alpha_k m)$ which is best matching $\rho_{\text{int},k}[m]$ in the interval $[M_k, M_0]$. Details and restrictions are outlined in Section F.2.2.*

5. *Set the new intermediate function to the difference between the old one and the fitted exponential: $\rho_{\text{int},k+1}[m]=\rho_{\text{int},k}[m] - \omega_k \exp(\alpha_k m)$.*

6. *If $M_k>1$, increment k and continue the recursion with Step 3; otherwise, end the iterations with Step 7.*

7. *Set the order K to $K=k+1$, $\alpha_K=-\infty$ and $\omega_K=1-\sum_{k=1}^{K-1}\omega_k$; in order to satisfy Eq. (F.10b).*

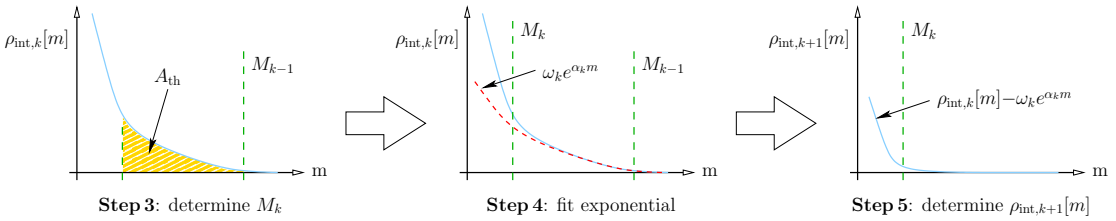
An overview of the recursive steps of the fitting procedure is given in Figure F.2. Respective description are provided in the following sections; thereby, the differences to known methods are highlighted as well.

F.2.1 Step 3: Determining the Border M_k

In order to capture LRD effects, it is of advantage to ensure a roughly logarithmic spacing between the borders of the fitting intervals M_k; in other words, $M_k \approx \frac{M_{k-1}}{\Delta_M}$, where $\Delta_M>1$ denotes a constant factor. Such a placement of the interval borders is proposed by [46].
In this work I propose to allow for a certain variability of Δ_M within each iteration, in order to avoid that exponentials are fitted to intervals where $\rho_{\text{int},k}[m]$ is close to zero. This can be achieved by ensuring that the area A_k below the intermediate function between M_{k-1} and M_k, exceeds a certain threshold value A_{th}. This area is calculated according to

$$A_k[m] = \sum_{l=m}^{M_{k-1}} |\rho_{\text{int},k}[l]| \frac{1}{l}, \tag{F.11}$$

which can be computed recursively for $m = M_{k-1}, \ldots, 0$, starting with the largest value M_{k-1}. Thereby, the division by l preserves the logarithmic spacing mentioned above; e.g., a constant offset in the interval $m=M_k, \ldots, \Delta_M M_k$ has the same area $A_k[m]$ as it would have in the interval $m = \frac{M_k}{\Delta_M}, \ldots, M_k$. The boundary M_k is fixed at the highest value where Eq. (F.11) yields $A_k[m]\geq A_{\text{th}}$.
In addition, a minimum distance $\Delta_{M,\min}$ between M_{k-1} and M_k is required, in order to avoid exponentials with similar values α_{k-1} and α_k. Such values would cause difficulties for the numerical solution of Eq. (F.9). Further, without a minimum distance, the



Step 3: determine M_k **Step 4**: fit exponential **Step 5**: determine $\rho_{\text{int},k+1}[m]$

FIGURE F.2: The three most important steps of the fitting procedure.

algorithm would be sensitive to measurement outliers. Empirical trials suggest that $\Delta_{M,\min}$=4 yields satisfactory results. Summarizing, M_k is selected according to

$$M_k = \max_{m\in\{0,\dots,M_{k-1}\}}(m) \qquad \text{subject to} \quad m\leq\frac{M_{k-1}}{\Delta_{M,\min}},\ A_k[m]\geq A_{\text{th}}. \tag{F.12}$$

The selection of the value A_{th} is not trivial; i.e., it cannot be assessed a priori which value will result in the best fitting accuracy. Therefore, I suggest to perform the whole fitting procedure several times with different threshold areas A_{th}. This method works very well, as reported in Section F.3.

F.2.2 Step 4: Fitting the Exponential

Fitting a single exponential $\omega_k e^{\alpha_k m}$ to the function $\rho_{\text{int},k}[l]$ within the interval $[M_k, M_{k-1}]$, yields three parameters to be optimized: ω_k, $\mathfrak{Re}\{\alpha_k\}=\alpha_{k,r}$ and $\mathfrak{Im}\{\alpha_k\}=\alpha_{k,i}$. However, since the α_k must occur in conjugate complex pairs in order to yield a real function, it is recommended to directly fit two exponentials of the form

$$\rho_{\text{int},k}[m] \approx \omega'_k\left(\exp\Big(\left(\alpha'_{k,r}+j\alpha'_{k,i}\right)m\Big)+\exp\Big(\left(\alpha'_{k,r}-j\alpha'_{k,i}\right)m\Big)\right), \tag{F.13}$$

where $j=\sqrt{-1}$. Thereby the actual work differs from [46], where only real values for α_k are considered. The reason is the strict monotonicity of CDFs to be fitted in [46], which makes any oscillations caused by the complex part of α_k obsolete. In the present work, where ACFs shall be modeled, the capability of modeling such oscillations is beneficial (cf. Section F.3).

For the actual fitting I propose to use the well-known Non-linear Least Squares (NLS) technique. This problem can be solved by iterative optimization methods [253], similar to Eq. (F.9). An adequate solution to this problem is usually found very fast, due to the low dimensionality. A plethora of software tools is available, capable of performing this task. Compared to related work, where only the two points $\rho_{\text{int},k}[M_k]$ and $\rho_{\text{int},k}[M_{k-1}]$ are involved by the fitting procedure, the presented algorithm is more robust at the presence of outliers and noise, since it involves all points in the fitting interval.

The initialization of the NLS algorithm, as well as the bounding of the output values is essential for the convergence of the NLS algorithm. Consider an arbitrary iteration step k, where Eq. (F.13) shall be fitted to $\rho_{\text{int},k}[l]$, then the following settings shall be provided to the NLS algorithm:

- The points of support are $\rho_{\text{int},k}[m]$ (y-axis) with $m=M_k,\dots,M_0$ (x-axis).
- The lower and upper bounds for ω'_k are 0 and $\frac{1}{2}\ (1-\sum_{l=1}^{k-1}\omega_l)$, respectively. The upper bound is required to satisfy Eq. (F.10b).
- The lower bound for $\alpha'_{k,r}$ is a value from $[-5,-1]$ (recommended: -1), since any smaller value causes the exponential to decay so fast that it yields roughly zero at lag m=1. The upper bound is a value between $[-\frac{1}{M_0},0[$ (recommended: $-\frac{1}{100\,M_0}$), which guarantees that Eq. (F.10a) is satisfied.
- The lower and upper bounds for $\alpha'_{k,i}$ are 0 and π, respectively. Negative values are covered anyways by the conjugate complex and higher values would cause aliasing effects (i.e., the respective oscillations cannot be resolved by $m\in\mathbb{Z}$).

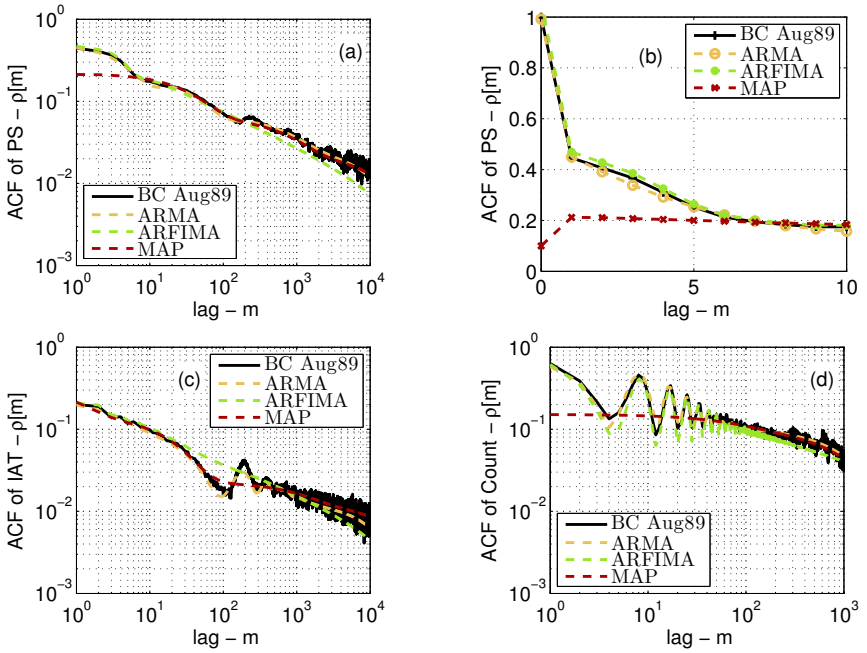


FIGURE F.3: Fitting the ACFs of three quantities (i.e., PS: (a–b), IAT: (c), packet counts: (d)) of the *Bellcore Aug89* trace. Different fitting methods: ARMA, ARFIMA [198] and MAP [261].

- The initial values for ω'_k and $\alpha'_{k,r}$ can be chosen randomly between the bounds; values close to zero are recommended.
- The initial value for $\alpha'_{k,i}$ is very sensitive. I recommend to choose it as the frequency f of the maximum value of the spectrum $P_{\text{int},k}(f)=\mathcal{F}(\rho_{\text{int},k}[m])$. This corresponds to $\alpha'_{k,i}=\arg\max_f P_{\text{int},k}(f)$.

Each recursion results in a sum of two complex conjugate exponentials. If the imaginary values $\pm\alpha'_{k,i}$ are thereby close to zero, the sum can be condensed to a single exponential: $\omega_k=2\omega'_k$ and $\alpha_k=\alpha'_{k,r}$. Otherwise, two exponentials have to be added to the model according to: $\omega_k=\omega'_k$, $\omega_{k+1}=\omega'_k$ and $\alpha_k=\alpha'_{k,r}+j\alpha'_{k,i}$, $\alpha_{k+1}=\alpha'_{k,r}-j\alpha'_{k,i}$. In this case the iteration counter k has to be incremented; including the updates $M_{k+1}=M_k$ and $\rho_{\text{int},k+1}[m]=\rho_{\text{int},k}[m]-\omega_k e^{\alpha_k m}$.

F.3 Performance Evaluation

The performance of the outlined approach is affirmed by Figure F.3 and Figure F.4. Both figures show measured network traffic traces modeled by three different procedures:

- **ARMA**: Denotes the present approach.
- **ARFIMA**: The approach presented in [198] has been implemented.
- **MAP**: The algorithm provided by [261] has been deployed.

In order to obtain a fair comparison, the number of model parameters of the three approaches is kept roughly constant. Thereby, the ARMA order was upper bounded by ARMA(5,5), the ARFIMA order was constant with ARFIMA(7,1,7) and the number of parameters deployed for the MAP process was upper bounded by 28 (automatic

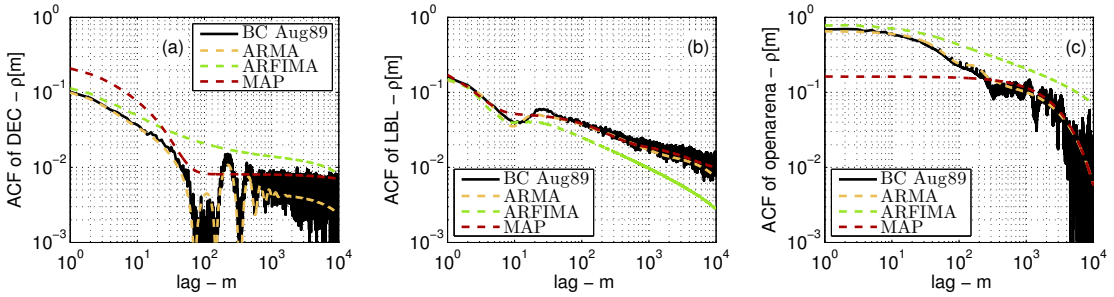


FIGURE F.4: Fitting the ACFs of the *DEC-PKT-1-UDP* (a), *LBL-PKT-5* (b) and *openarena* (c) traces. Different methods: ARMA, ARFIMA [198] and MAP [261].

optimization). The time required for model fitting was similar for all three approaches (i.e., roughly 1 min for fitting 10^6 samples). The complexity of generating samples from the models is $\mathcal{O}(N)$ in the case of ARMA and MAP processes, whereas it is $\mathcal{O}(N \log(N)$ for the ARFIMA process, due to the required fractional integration.

The traces used for the performance evaluation are *Bellcore Aug89*, *DEC-PKT-1-UDP* and *LBL-PKT-5* released in the context of [197] and available at [202]. Those traces have been deployed for performance evaluation of traffic models by many authors (cf. [180] [258]) and are therefore also used in the present work. Further, a trace of the online game *openarena* has been measured and used for evaluation as well.

In Figure F.3 the *Bellcore Aug89* trace is extensively examined. Various quantities are thereby considered, namely, (i) the PS in Figure F.3 (a–b) in linear and logarithmic scale, (ii) the IAT in Figure F.3 (c) and (iii) the packet counts per 100 ms time interval in Figure F.3 (d). In Figure F.4 the other three traces are evaluated in the same way. The proposed method (denoted as ARMA) shows very promising accuracy, for all evaluated scenarios. Notice, that oscillations in the ACF can easily be resembled, whereas the other modeling approaches have difficulties in that case. The ARMA approach performs slightly better than both other modeling approaches; the risk of obtaining a poor model is significantly reduced. Thus, the proposed method is well-suited for fully automated fitting procedures.

Appendix G

TARMA Model Parameters for Specific Applications

In the following the model parameters are listed, which are obtained from the Transformed Auto-Regressive Moving-Average (TARMA) modeling procedure presented in Section 5.1. Note, however, that the parameters are very sensitive to rounding errors, it is therefore recommended to download the exact model parameters at [121].
In order to generate synthetic traffic from these values, the following procedure has to be carried out. Examples are given in the *Matlab* programming language.

- Parameter $g_{21}[m]$ has to be embedded into $\mathbf{G}$ according to Eq. (5.19), yielding matrix `G`.
- The polynomials $\theta(B)$ of both random processes (Packet Size (PS) and Inter Packet-Arrival Time (IAT)) have to be embedded into vectors of filter coefficients `b_ps` and `b_iat` (direct mapping).
- The polynomials $\phi(B)$ of both random processes (PS and IAT) have to be embedded into vectors of filter coefficients `a_ps` and `a_iat` (direct mapping).
- The coefficients of both polynomials $\mathfrak{p}(y)$ have to be mapped to vectors of polynomial coefficients `p_ps` and `p_iat`
- Having performed this mapping, synthetic samples are generated by:
  - `x    = G * randn(2,N);`
  - `y_ps = filter( b_ps, a_ps, x(1,:));`
  - `z_ps = polyval( p_ps, y_ps);`

  Thereby, `N` denotes the number of requested samples. The last two steps have further to be carried out for `y_iat` and `z_iat`, where the respective coefficient vectors have to be deployed and `x(2,:)` is input.

This extremely simple procedure for the generation of synthetic samples, combined with the parsimonious representation of complex traffic streams, makes the TARMA model a promising alternative to state of the art modeling approaches.

order	5	4	3	2	1	0	unit
$\mathfrak{p}_{\mathrm{PS}}(y)$	-4.6	-77.6	9.7	398	434	201	byte
$\phi_{\mathrm{PS}}(B)$	-0.68	3.7	-7.9	8.5	-4.6	1	
$\theta_{\mathrm{PS}}(B)$	0.8	-3.4	5.8	-4.9	2	-0.34	
$\mathfrak{p}_{\mathrm{IAT}}(y)$	0.19	1	-0.26	-1.34	1.78	2.41	ms
$\phi_{\mathrm{IAT}}(B)$	-0.88	4.5	-9.2	9.5	-4.9	1	
$\theta_{\mathrm{IAT}}(B)$	0.88	-4.1	7.8	-7.3	3.4	-0.63	
$g_{21}[m]$						-0.13	

TABLE G.1: Model parameters *Bellcore Aug89* [202], cf. Figure 5.8

order	5	4	3	2	1	0	unit
$\mathfrak{p}_{\mathrm{PS}}(y)$	1.8	18.6	-1.9	-11.8	67.2	157	byte
$\phi_{\mathrm{PS}}(B)$				0.971	-1.971	1	
$\theta_{\mathrm{PS}}(B)$				0.44	-1.04	0.6	
$\mathfrak{p}_{\mathrm{IAT}}(y)$						40	ms
$g_{21}[m]$						0.41	

TABLE G.2: Model parameters *openarena* [8], cf. Figure 5.9

order	5	4	3	2	1	0	unit
$\mathfrak{p}_{\mathrm{I}}(y)$	0.074	-0.8	-0.1	10.4	32.3	55.4	kbyte
$\mathfrak{p}_{\mathrm{P}}(y)$	0.02	-0.13	-0.74	7.15	19.1	20.9	kbyte
$\mathfrak{p}_{\mathrm{B}}(y)$	-0.002	-0.09	0.7	4.32	10.3	11.7	kbyte
$\phi_{\mathrm{I,P,B}}(B)$				0.984	-1.984	1	
$\theta_{\mathrm{I,P,B}}(B)$				0.358	-0.829	0.47	
$\mathfrak{p}_{\mathrm{IAT}}(y)$ - ms						40	

TABLE G.3: Model parameters *Lord of the Rings I* [204], cf. Figure 5.10

Appendix H

Definition of Distributions

Name	Symbol	PDF(x)[1] or PMF(k)[2]	Support
Normal	$\mathcal{N}(\mu, \sigma)$	$\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$	$\mathbb{R}$
Uniform	$\mathcal{U}(L, H)$	$\frac{1}{H-L}$	$[H, L]$
Binomial	$\mathcal{B}(n, p)$	$\binom{n}{k}\ p^k\ (1-p)^{n-k}$	$\{0, 1, \ldots, n\}$
Poisson	$\mathcal{P}ois(\lambda)$	$\frac{\lambda^k}{k!}\ e^{-\lambda}$	$\mathbb{N} \cup \{0\}$
Geometric	$\mathcal{G}eom(p)$	$(1-p)^k\ p$	$\mathbb{N} \cup \{0\}$
Exponential	$\mathcal{E}xp(\lambda)$	$\lambda\ \exp(-\lambda x)$	$[0, \infty]$
Gamma	$\mathcal{G}am(\alpha, \theta)$	$\frac{1}{\Gamma(\alpha)\ \theta^\alpha}\ x^{\alpha-1}\ \exp\left(-\frac{x}{\theta}\right)$	$[0, \infty]$
Weibull	$\mathcal{W}bl(\theta, \alpha)$	$\frac{\alpha}{\theta}\ \left(\frac{x}{\theta}\right)^{\alpha-1}\ \exp\left(-\frac{x}{\theta}\right)^\alpha$	$[0, \infty]$
Log-Normal	$\mathcal{L}ogN(\mu, \sigma)$	$\frac{1}{x\ \sqrt{2\pi\sigma^2}} \exp\left(-\frac{(\ln(x)-\mu)^2}{2\sigma^2}\right)$	$[0, \infty]$
Pareto	$\mathcal{P}(\alpha, L)$	$\frac{\alpha\ L^\alpha}{x^{\alpha+1}}$	$[L, \infty]$
Bounded Pareto	$\mathcal{BP}(\alpha, L, H)$	$\frac{\alpha\ L^\alpha\ x^{-\alpha-1}}{1-(L/H)^\alpha}$	[L,H]
Generalized Pareto	$\mathcal{GP}(\xi, \sigma, L)$	$\frac{1}{\sigma}\left(1 + \xi\ \frac{x-L}{\sigma}\right)^{-(1+1/\xi)}$	$[L, \infty]$
Beta[3]	$\mathcal{B}eta(\alpha, \beta)$	$\frac{x^{\alpha-1}\ (1-x)^{\beta-1}}{B(\alpha,\beta)}$	[0,1]

[1] Probability Density Function (PDF), continuous support x.
[2] Probability Mass Function (PMF), discrete support k.
[3] $B(\cdot, \cdot)$ denotes the beta-function

Bibliography

[1] M. Laner, P. Svoboda, and M. Rupp, "Dissecting 3G Uplink Delay by Measuring in an Operational HSPA Network," in *PAM'11, Atlanta, Georgia*, 2011.

[2] M. Laner, P. Svoboda, and M. Rupp, "Latency Analysis of 3G Network Components," in *EW'12, Poznan, Poland*, 2012.

[3] 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 *WiNMee'12, Paderborn, Germany*, 2012.

[4] P. Svoboda, M. Laner, J. Fabini, M. Rupp, and F. Ricciato, "Packet Delay Measurements in Reactive IP Networks," *IEEE Instrum. Meas. Mag.*, vol. 15(6), pp. 36–43, 2012.

[5] M. Laner, J. Fabini, P. Svoboda, and M. Rupp, "End-to-end Delay in Mobile Networks: Does the Traffic Pattern Matter?," in *ISWCS'13, Ilmenau, Germany*, 2013.

[6] M. Laner, S. Caban, P. Svoboda, and M. Rupp, "Time Synchronization Performance of Desktop Computers," in *ISPCS'11, Munich*, 2011.

[7] M. Laner, P. Svoboda, and M. Rupp, "A Benchmark Methodology For End-to-End Delay of Reactive Mobile Networks," in *WD'13, Valencia, Spain*, 2013.

[8] M. Laner, P. Svoboda, and M. Rupp, "Modeling Randomness in Network Traffic," in *SIGMETRICS'12, London, UK*, 2012.

[9] 3GPP, "TR 36.822, LTE RAN enhancements for diverse data applications." [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/36822.htm.

[10] M. Laner, P. Svoboda, N. Nikaein, and M. Rupp, "Traffic Models for Machine Type Communications," in *ISWCS'13, Ilmenau, Germany*, 2013.

[11] M. Laner, P. Svoboda, S. Schwarz, and M. Rupp, "Users in Cells: a Data Traffic Analysis," in *WCNC'12, Paris, France*, 2012.

[12] EU FP 7 LOLA Project, "Work Package 3, Traffic Measurement and Modelling, Deliverables 3.1–3.6," 2013. [Online]. Available: http://www.ict-lola.eu/deliverables/wp3-traffic-measurement-and-modelling.

[13] V. Paxson *et al.*, "RFC 2330, Framework for IP Performance Metrics," 1998. [Online]. Available: www.ietf.org/rfc/rfc2330.txt.

[14] B. A. Forouzan, *Data Communications and Networking*. McGraw - Hill, 2007.

[15] ITU-T, "X.200: Information technology –Open Systems Interconnection – Basic Reference Model: The basic model," 1994. [Online]. Available: www.itu.int/rec/T-REC-X.200-199407-I/en.

[16] G. Almes, S. Kalidindi, and M. Zekauskas, "RFC 2679, A One-way Delay Metric for IPPM," 1999. [Online]. Available: www.ietf.org/rfc/rfc2679.txt.

[17] G. Almes, S. Kalidindi, and M. Zekauskas, "RFC 2681, A Round-trip Delay Metric for IPPM," 1999. [Online]. Available: www.ietf.org/rfc/rfc2681.txt.

[18] C. Demichelis and P. Chimento, "RFC 3393, IP Packet Delay Variation Metric for IPPM," 2002. [Online]. Available: www.ietf.org/rfc/rfc3393.txt.

[19] V. Raisanen, G. Grotefeld, and A. Morton, "RFC 3432, Network performance measurement with periodic streams," 2002. [Online]. Available: www.ietf.org/rfc/rfc3432.txt.

[20] ITU-T, "I.380: IP packet transfer and availability performance parameters," 1999. [Online]. Available: www.itu.int/rec/T-REC-I.380-199902-S/en.

[21] 3GPP, "TS 25.913, Requirements for Evolved UTRA and Evolved UTRAN." [Online]. Available: www.3gpp.org.

[22] H. Holma and A. Toskala, *WCDMA for UMTS: HSPA Evolution and LTE (5th Edition)*. Wiley, 2010.

[23] D. P. Bertsekas and R. Gallager, *Data Networks, 2nd Edition*. Prentice Hall, 1992.

[24] R. Nelson, *Probability, Stochastic Processes, and Queueing Theory*. Springer, 1995.

[25] A. Pasztor and D. Veitch, "On the Scope of End-to-End Probing Methods," *IEEE Commun. Lett.*, vol. 6(11), pp. 509–511, 2002.

[26] K. Lai and M. Baker, "Measuring Link Bandwidths Using a Deterministic Model of Packet Delay," in *SIGCOMM'00, Stockholm, Sweden*, 2000.

[27] J. Medhi, *Stochastic Models in Queueing Theory, Second Edition*. Academic Press, 2002.

[28] F. Baccelli and P. Bremaud, *Elements of Queueing Theory, 2nd Edition*. Springer, 2003.

[29] M. Zuckermann, "Introduction to Queueing Theory and Stochastic Teletraffic Models." Lecture notes, [Online]. Available: www.ee.cityu.edu.hk/~zukerman/classnotes.pdf, 2011.

[30] V. Frost and B. Melamed, "Traffic Modeling For Telecommunication Networks," *IEEE Commun. Mag.*, vol. 32 (3), pp. 70–81, 1994.

[31] D. M. Lucantoni, "New Results On The Single Server Queue With A Batch Markovian Arrival Process," *Stochasitc Models*, vol. 7 (1), pp. 1–46, 1991.

[32] P. Svoboda, *Measurement and Modelling of Internet Traffic over 2.5 and 3G Cellular Core Networks*. PhD thesis, Vienna University of Technology, 2008. [Online]. Available: publik.tuwien.ac.at/files/PubDat_170755.pdf.

[33] D. G. Kendall, "Stochastic Processes Occurring in the Theory of Queues and their Analysis by the Method of the Imbedded Markov Chain," *Annals Math. Statistics*, vol. 24 (3), pp. 338–354, 1953.

[34] M. Aida, N. Miyoshi, and K. Ishibashi, "A Change-of-Measure Approach of Per-Flow Delay Measurement Combining Passive and Active Methods: Mathematical Formulation of the CoMPACT Monitor," *IEEE Trans. Inf. Theory*, vol. 54(11), pp. 4966–4979, 2008.

[35] J. D. C. Little, "A Proof for the Queuing Formula: $L{=}\lambda W$," *Operations Research*, vol. 9 (3), pp. 383–320, 1961.

[36] R. Wolff, "Poisson Arrivals See Time Averages," *Operations Research*, vol. 30 (2), pp. 223–231, 1982.

[37] B. Melamed and W. Whitt, "On Arrivals That See Time Averages," *Operations Research*, vol. 38 (1), pp. 156–172, 1990.

[38] M. Tariq *et al.*, "Poisson versus periodic probing (or, does PASTA matter?)," in *IMC'05, Berkeley, CA*, 2005.

[39] F. Baccelli *et al.*, "On Optimal Probing for Delay and Loss Measurement," in *IMC'07, San Diego, California*, 2007.

[40] M. Roughan, "A Comparison of Poisson and Uniform Sampling for Active Measurements," *IEEE J. Sel. Areas Commun.*, vol. 24 (12), pp. 2299–2312, 2006.

[41] T. Lindh, "Systematic Sampling and Cluster Sampling of Packet Delays," in *PAM'06, Adelaide, Australia*, 2006.

[42] S. Machiraju, D. Veitch, F. Baccelli, and J. Bolot, "Adding Definition to Active Probing," *ACM Computer Commun. Rev.*, vol. 37(2), pp. 19–28, 2007.

[43] Baccelli *et al.*, "The Role of PASTA in Network Measurements," in *SIGCOMM'06, Pisa, Italy*, 2006.

[44] T. Zseby, "Deployment of Sampling Methods for SLA Validation with Non-Intrusive Measurements," in *PAM'02, Fort Collins, Colorado*, 2002.

[45] F. Baccelli, B. Kauffmann, and D. Veitch, "Inverse problems in queueing theory and Internet probing," *Springer Queueing Syst.*, vol. 63, pp. 59–107, 2009.

[46] A. Feldmann and W. Whitt, "Fitting mixtures of exponentials to long-tail distributions to analyze network performance models," *Elsevier Perform. Eval.*, vol. 31(3), pp. 245–279, 1998.

[47] K. Lakshminarayanan, V. N. Padmanabhan, and J. Padhye, "Bandwidth Estimation in Broadband Access Networks," in *IMC'04, Taormina, Italy*, 2004.

[48] R. Prasad *et al.*, "Bandwidth estimation: metrics, measurement techniques, and tools," *IEEE Network*, vol. 17(6), pp. 27–35, 2003.

[49] R. Lubben, M. Fidler, and J. Liebeherr, "A Foundation for Stochastic Bandwidth Estimation of Networks with Random Service," in *INFOCOM'11, Shanghai, China*, 2011.

[50] E. Lawrence, G. Michailidis, and V. N. Nair, "Statistical Inverse Problems in Active Network Tomography," *IMS Lecture Notes-Monogr.*, vol. 54, pp. 24–44, 2007.

[51] ITU-R, "TF.460: Standard-frequency and time-signal emissions," 2002. [Online]. Available: www.itu.int/rec/R-REC-TF.460-6-200202-I/en.

[52] D. Mills *et al.*, "RFC 5905, Network Time Protocol Version 4: Protocol and Algorithms Specification," 2010. [Online]. Available: www.ietf.org/rfc/rfc5905.txt.

[53] O. Gurewitz and M. Sidi, "Estimating One-Way Delays From Cyclic-Path Delay Measurements," in *Infocom'01, Anchorage, AK*, 2001.

[54] O. Gurewitz, I. Cidon, and M. Sidi, "One-Way Delay Estimation Using Network-Wide Measurements," *IEEE Trans. Inf. Theory*, vol. 52 (6), pp. 2710–2724, 2006.

[55] A. Vakili and J. Grgoire, "Accurate One-Way Delay Estimation: Limitations and Improvements," *IEEE Trans. Instrum. Meas.*, vol. 61 (9), pp. 2428–2435, 2012.

[56] F. Vacirca, F. Ricciato, and R. Pilz, "Large-Scale RTT Measurements from an Operational UMTS/GPRS Network," in *WICON'05, Budapest, Hungary*, 2005.

[57] P. Romirer *et al.*, "Network-Wide Measurements of TCP RTT in 3G," in *TMA'09, Aachen, Germany*, 2009.

[58] A. Gember *et al.*, "Obtaining In-Context Measurements of Cellular Network Performance," in *IMC'12, Boston, MA*, 2012.

[59] L. Schumacher, G. Gomand, and G. Toma, "Performance Evaluation of Indoor Internet Access over a Test LTE Mini-Network," in *WPMC'11, Brest, France*, 2011.

[60] M. Wylie-Green and T. Svensson, "Throughput, Capacity, Handover and Latency Performance in a 3GPP LTE FDD Field Trial," in *Globecom'10, Miami, Florida*, 2010.

[61] J. Fabini, P. Reichl, and A. Poropatich, "A Generic Approach to Access Network Modeling for Next Generation Network Applications," in *ICNS'08, Gosier, Guadeloupe*, 2008.

[62] P. Arlos and M. Fiedler, "Influence of the Packet Size on the One-Way Delay in 3G Networks," in *PAM'10, Zurich, Switzerland*, 2010.

[63] P. Arlos and M. Fiedler, "Influence of the packet size on the one-way delay on the down-link in 3G networks," in *ISWPC'10, Modena, Italy*, 2010.

[64] B.-K. Choi *et al.*, "Analysis of point-to-point packet delay in an operational network," in *INFOCOM'04, Hong Kong*, 2004.

[65] M. Jurvansuu, J. Prokkola, M. Hanski, and P. Perala, "HSDPA Performance in Live Networks," in *ICC'07, Glasgow, Scotland*, 2007.

[66] J. Prokkola *et al.*, "Measuring WCDMA and HSDPA Delay Characteristics with QoSMeT," in *ICC'07, Glasgow, Scotland*, 2007.

[67] F. Ricciato, E. Hasenleithner, and P. Romirer-Maierhofer, "Traffic analysis at short time-scales: an empirical case study from a 3G cellular network," *IEEE Trans. Netw. Service Manag.*, vol. 5 (1), pp. 11–21, 2008.

[68] W. John, S. Tafvelin, and T. Olovsson, "Passive internet measurement: Overview and guidelines based on experiences," *Elsevier Computer Commun.*, vol. 33, pp. 533–550, 2010.

[69] J. Fabini, *Generic Access Network Modeling for Next Generation Network Applications*. PhD thesis, Vienna University of Technology, 2008.

[70] M. Shin *et al.*, "Survey on the Clock Synchronization Schemes for Propagation Delay Measurement," *Int. J. Adv. Science Techn.*, vol. 35, pp. 139–150, 2011.

[71] V. Paxson, "On Calibrating Measurements of Packet Transit Times," in *SIGMETRICS/Performance'98, Madison, WI*, 1998.

[72] S. Moon, P. Skelly, and D. Towsley, "Estimation and Removal of Clock Skew from Network Delay Measurements," in *INFOCOM'99, New York, NY*, 1999.

[73] L. Zhang, Z. Liu, and C. H. Xia, "Clock Synchronization Algorithms for Network Measurements," in *INFOCOM'02, New York, NY*, 2002.

[74] J. Wang, M. Zhoua, and H. Zhoub, "Clock synchronization for Internet measurements: a clustering algorithm," *Elsevier Computer Netw.*, vol. 45 (6), pp. 731–741, 2004.

[75] J. Bi, Q. Wu, and Z. Li, "On estimating clock skew for one-way measurements," *Elsevier Computer Commun.*, vol. 29, pp. 1213–1225, 2006.

[76] Y. Lin *et al.*, "A Fuzzy-Based Approach to Remove Clock Skew and Reset From One-Way Delay Measurement," *IEEE Trans. Neural Netw.*, vol. 16 (5), pp. 1125–1135, 2005.

[77] H. Khlifi and J. Grgoire, "Low-complexity offline and online clock skew estimation and removal," *Elsevier Computer Netw.*, vol. 50, pp. 1872–1884, 2006.

[78] D. Allan, N. Ashby, and C. Hodge, "The science of timekeeping," tech. rep., Hewlett Packard, 1997. [Online]. Available: www.allanstime.com/Publications/DWA/Science_Timekeeping/TheScienceOfTimekeeping.pdf.

[79] J. Levine, "Introduction to Time and Frequency Metrology," *Rev. Scientific Instrum.*, vol. 70, pp. 2567–2596, 1999.

[80] D. Mills, "The Network Computer as Precision Timekeeper," in *28th PTTI Meeting, Reston, VA*, 1996.

[81] D. Mills, "Internet Time Synchronization: The Network Time Protocol," *IEEE Trans. Commun.*, vol. 39 (10), pp. 1482–1493, 1991.

[82] D. Mills, "RFC 1305, Network Time Protocol Version 3: Specification, Implementation and Analysis," 1992. [Online]. Available: www.ietf.org/rfc/rfc1305.txt.

[83] D. Veitch, J. Ridoux, and S. Korada, "Robust Synchronization of Absolute and Difference Clocks Over Networks," *IEEE/ACM Trans. Netw.*, vol. 17 (2), pp. 417–430, 2009.

[84] T. Zseby, S. Zander, and G. Carle, "Evaluation of Building Blocks for Passive One-way-delay Measurements," in *PAM'01, Amsterdam, Netherlands*, 2001.

[85] L. De Vito, S. Rapuano, and L. Tomaciello, "One-Way Delay Measurement: State of the Art," *IEEE Trans. Instrum. Meas.*, vol. 57 (12), pp. 2742–2750, 2008.

[86] S. Donnelly, *High Precision Timeing in Passive Measurements of Data Networks*. PhD thesis, University of Waikato, Hamilton, New Zealand, 2002. [Online]. Available: wand.cs.waikato.ac.nz/old/wand/publications/stephen-thesis.pdf.

[87] "Endace DAG, network monitoring cards." [Online]. Available: www.endace.com.

[88] IEEE, "1588: Standard for A Precision Clock Synchronization Protocol for Networked Measurement and Control Systems." [Online]. Available: www.nist.gov/el/isd/ieee/ieee1588.cfm.

[89] J. C. Edison, *Measurement, Control and Communication Using IEEE 1588*. Springer, 2006.

[90] K. Correll, N. Barendt, and M. Branicky, "Design Consideration for Software Only Implementations of the IEEE 1588 Precision Time Protocol," in *Conf. on IEEE 1588, Zurich, Switzerland*, 2005.

[91] R. Cochran and C. Marinescu, "Design and Implementation of a PTP Clock Infrastructure for the Linux Kernel," in *ISPCS'10, Portsmouth, NH*, 2010.

[92] J. Nonaka *et al.*, "Low-Cost Hybrid Internal Clock Synchronization Mechanism for COTS PC Cluster," in *Euro-Par'02, Paderborn, Germany*, 2002.

[93] "LinuxPPS, a linux driver for PPS signals." [Online]. Available: wiki.enneenne.com/index.php/LinuxPPS_support.

[94] ITU-T, "G.8261/Y.1361: Timing and synchronization aspects in packet networks," 2008. [Online]. Available: www.itu.int/rec/T-REC-I.380-199902-S/en.

[95] P. Moreira *et al.*, "White Rabbit: Sub-Nanosecond Timing Distribution over Ethernet," in *ISPCS'09, Brescia, Italy*, 2009.

[96] F. Ricciato, "Traffic monitoring and analysis for the optimization of a 3G network," *EEE Trans. Wireless Commun.*, vol. 13, pp. 42–49, 2006.

[97] M. Rupp, ed., *Video and Multimedia Transmissions over Cellular Networks*. Wiley, 2009.

[98] K. Claffy, "Ten Things Lawyers should know about the Internet," tech. rep., CAIDA, SDSC, UCSD, 2008. [Online]. Available: www.caida.org/publications/papers/2008/lawyers_top_ten/lawyers_top_ten.pdf.

[99] D. Sicker, P. Ohm, and D. Grunwald, "Legal Issues Surronding Monitoring During Network Reseach," in *IMC'07, San Diego, CA*, 2007.

[100] A. Slagell, J. Wang, and W. Yurcik, "Network log anonymization: Application of crypto-pan to cisco netflows," in *SKM'04, Buffalo, NY*, 2004.

[101] C. Shannon *et al.*, "The Internet Measurement Data Catalog," *ACM Computer Commun. Rev.*, vol. 35(5), pp. 97–100, 2005.

[102] S. Coull *et al.*, "Playing Devil's Advocate: Inferring Sensitive Information from Anonymized Network Traces," in *NDSS'07, San Diego, CA*, 2007.

[103] M. Allman and V. Paxson, "Issues and Etiquette Concerning Use of Shared Measurement Data," in *IMC'07, San Diego, CA*, 2007.

[104] K. Chen, "Quadrant of Euphoria: A Crowdsourcing Platform for QoE Assessmen," *IEEE Network*, vol. 24(2), pp. 28–35, 2010.

[105] J. Sommers and P. Barford, "Cell vs. WiFi: On the Performance of Metro Area Mobile Connections," in *IMC'12, Boston, MA*, 2012.

[106] "Wireshark - network protocol analyzer." [Online]. Available: www.wireshark.org.

[107] T. Zseby *et al.*, "RFC 5472, IP Flow Information Export (IPFIX) Applicability," 2009. [Online]. Available: www.ietf.org/rfc/rfc5472.txt.

[108] V. Moreno *et al.*, "Batch to the Future: Analyzing Timestamp Accuracy of High-Performance Packet I/O Engines," *IEEE Commun. Lett.*, vol. 16(11), pp. 1888–1891, 2012.

[109] J. L. Garcia-Dorado *et al.*, *High-Performance Network Traffic Processing Systems Using Commodity Hardware*, ch. 1, pp. 3–27. Springer, 2013.

[110] "NetFPGA, network monitoring cards." [Online]. Available: http://netfpga.org.

[111] "Napatech, network monitoring cards." [Online]. Available: www.napatech.com.

[112] "Invea-Tech, network monitoring cards." [Online]. Available: www.invea-tech.com.

[113] S. Alcock, P. Lorier, and R. Nelson, "Libtrace: A Packet Capture and Analysis Library," *ACM Computer Commun. Rev.*, vol. 42(2), pp. 42–48, 2012.

[114] "libpcap, network traffic capture library." [Online]. Available: www.tcpdump.org.

[115] T. Zseby *et al.*, "RFC 5475, Sampling and Filtering Techniques for IP Packet Selection," 2009. [Online]. Available: www.ietf.org/rfc/rfc5475.txt.

[116] B. Claise, A. Johnson, and J. Quittek, "RFC 5476, Packet Sampling (PSAMP) Protocol Specifications," 2009. [Online]. Available: www.ietf.org/rfc/rfc5476.txt.

[117] M. Lee, N. Duffield, and R. R. Kompella, "MAPLE: A Scalable Architecture for Maintaining Packet Latency Measurements," in *IMC'12, Boston, MA*, 2012.

[118] G. E. P. Box, J. S. Hunter, and W. G. Hunter, *Statistics for experimenters. Design, innovation, and discovery. 2nd ed.* Wiley, 2005.

[119] D. C. Montgomery, *Design and Analysis of Experiments.* Wiley, 2009.

[120] A. M. Zoubir and D. R. Iskander, "Bootstrap Methods and Applications," *IEEE Signal Process. Mag.*, vol. 24(4), pp. 10–19, 2007.

[121] "Vienna University of Technology, Institute of Telecommunication – Downloads." [Online]. Available: http://www.nt.tuwien.ac.at/about-us/staff/markus-laner/.

[122] "Linear Technology, LT5534 – RF Power Detector." [Online]. Available: www.linear.com.

[123] 3GPP, "TS 25.402, Synchronisation in UTRAN Stage 2." [Online]. Available: www.3gpp.org.

[124] "Darwin Project." [Online]. Available: userver.ftw.at/~ricciato/darwin.

[125] M. Laner, P. Svoboda, and M. Rupp, "Outer-Loop Power Control in a live UMTS network: Measurement, Analysis and Improvements ," in *ISCCSP'10, Limassol, Cyprus*, 2010.

[126] M. Laner, P. Svoboda, and M. Rupp, "Measurement Aided Model Design for WCDMA Link Error Statistics," in *ICC'11, Kyoto, Japan*, 2011.

[127] J. Fabini and A. Morton, "Draft: Advanced Stream and Sampling Framework for IPPM," 2012. [Online]. Available: tools.ietf.org/pdf/draft-morton-ippm-2330-update-00.pdf.

[128] J. Fabini *et al.*, "The Illusion of Being Deterministic Application-Level Considerations on Delay in 3G HSPA Networks," in *NETWORKING'09*, Springer, 2009.

[129] J. Fabini, L. Wallentin, and P. Reichl, "The importance of being really random: methodological aspects of IP-layer 2G and 3G network delay assessment," in *ICC'09, Dresden, Germany*, 2009.

[130] G. Gan, C. Ma, and J. Wu, *Data Clustering: Theory, Algorithms, and Applications*. SIAM, 2007.

[131] H. Cramer, "On the composition of elementary errors," *Scandinavian Actuarial Journal*, vol. 1928, no. 1, pp. 13–74, 1928.

[132] T. W. Anderson, "On the Distribution of the Two-Sample Cramer-von Mises Criterion," *Ann. Math. Statist.*, vol. 33(3), pp. 1148–1159, 1962.

[133] H. Holma and A. Toskala, *LTE for UMTS, OFDMA and SC-FDMA Based Radio Access*. Wiley, 2009.

[134] A. Hafsaoui, N. Nikaein, and C. Bonnet, "Analysis and Experimentation with a Realistic Traffic Generation Tool for Emerging Application Scenarios," in *EMU-Tools'13, Cannes, France*, 2013.

[135] 3GPP, TSG-RAN WG2#53, "R2-061402, Concept evaluation of user plane latency in LTE," 2006. Shanghai, China, [Online]. Available: www.3gpp.org/ftp/tsg_ran/WG2_RL2/TSGR2_53/Documents/R2-061402.zip.

[136] T. Blajic, D. Nogulic, and M. Druzijanic, "Latency Improvements in 3G Long Term Evolution," in *MIPRO'07, Opatija, Croatia*, 2007.

[137] N. S. Networks, "The Impact of latency on application performance," tech. rep., NSN, 2009. [Online]. Available: www.nokiasiemensnetworks.com/system/files/document/LatencyWhitepaper.pdf.

[138] D. Singhal, M. Kunapareddy, and V. Chetlapalli, "LTE-Advanced: Latency Analysis for IMT-A Evaluation," tech. rep., Tech Mahindra, 2010. [Online]. Available: www.techmahindra.com/Documents/WhitePaper/White_Paper_Latency_Analysis.pdf.

[139] S. Mohan, R. Kapoor, and B. Mohanty, "Latency in HSPA Data Networks," tech. rep., Qualcomm, 2011. [Online]. Available: www.qualcomm.com/media/documents/files/qualcomm-research-latency-in-hspa-data-networks.pdf.

[140] J. Liu, P. Tapia, P. Kwok, and Y. Karimli, "Performance and Capacity of HSUPA in Lab Environment," in *VTC Spring 2008, Singapore*, 2008.

[141] J. Robson, "The LTE/SAE Trail Initiative: Taking LTE/SAE from Specification to Rollout," *IEEE Communications Magazine*, vol. 47 (4), pp. 82–88, 2009.

[142] C. Serrano, B. Garriga, J. Velasco, J. Urbano, S. Tenorio, and M. Sierra, "Latency in Broad-Band Mobile Networks," in *VTC Spring 2009, Barcelona, Spain*, 2009.

[143] J. Prokkola *et al.*, "3G/HSPA Performance in Live Networks from the End User Perspective," in *ICC'09, Dresden, Germany*, 2009.

[144] D. Kim *et al.*, "Performance Measurement over Mobile WiMAX/IEEE 802.16e Network," in *WoWMoM'08, Newport Beach, CA*, 2008.

[145] J. Fabini and M. Abmayer, "Delay Measurement Methodology Revisited: Time-Slotted Randomness Cancellation," *IEEE Trans. Instrum. Meas.*, vol. 62(10), pp. 2839–2848, 2013.

[146] J. Fabini, M. Hirschbichler, J. Kuthan, and W. Wiedermann, "Mobile SIP: An Empirical Study on SIP Retransmission Timers in HSPA 3G Networks," in *EUNICE'13, Chemnitz, Germany*, 2013.

[147] P. Romirer, A. Coluccia, and T. Witek, "On the Use of TCP Passive Measurements for Anomaly Detection: A Case Study from an Operational 3G Network," in *TMA'10, Zurich, Switzerland*, 2010.

[148] P. Romirer, F. Ricciato, and A. Coluccia, "Explorative analysis of one-way delays in a mobile 3G network," in *LANMAN'08, Cluj-Napoca, Romania*, 2008.

[149] P. Romirer, F. Ricciato, and A. Coluccia, "Towards Anomaly Detection in One-Way Delay Measurements for 3G Mobile Networks: A Preliminary Study," in *IEEE/IFIP IPOM'08, Samos Island, Greece*, 2008.

[150] 3GPP, "TS 25.214, Physical layer procedures." [Online]. Available: www.3gpp.org.

[151] A. Hafsaoui, N. Nikaein, and L. Wang, "OpenAirInterface Traffic Generator (OTG): A Realistic Traffic Generation Tool for Emerging Application Scenarios ," in *MASCOTS'12, Washington, DC*, 2012.

[152] A. Adas, "Traffic Models in Broadband Networks," *IEEE Commun. Mag.*, vol. 35(7), pp. 82–89, 1997.

[153] H. Che and S. Li, "Fast Algorithm for Measurement- Based Traffic Modeling," *IEEE J. Sel. Areas Commun.*, vol. 16(5), pp. 612–625, 1998.

[154] S. Tanwir and H. Perros, "A Survey of VBR Video Traffic Models," *IEEE Commun. Surv. Tutor.*, vol. PP(99), pp. 1–25, 2013.

[155] Q. Zhang, "A General AR-Based Technique for the Generation of Arbitrary Gamma VBR Video Traffic in ATM Networks," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9(7), pp. 1130–1137, 1999.

[156] J. Faerber, "Network Game Traffic Modeling," in *NetGames'02, Braunschweig, Germany*, 2002.

[157] M. Livny, B. Melamed, and A. Tsiolis, "The Impact of Autocorrelation on Queuing Systems," *Management Science*, vol. 39(6), pp. 322–339, 1993.

[158] B. Hajek and L. He, "On Variations of Queue Response for Inputs with the Same Mean and Autocorrelation Function," *IEEE/ACM Trans. Netw.*, vol. 6(5), pp. 588–598, 1998.

[159] S. Li and C. Hwang, "Queue Response to Input Correlation Functions: Discrete Spectral Analysis," *IEEE/ACM Trans. Netw.*, vol. 1(5), pp. 678–692, 1993.

[160] A. Andersen and B. Nielsen, "On the Use of Second-order Descriptors to Predict Queueing Behavior of MAPs," *Naval Research Logistics (NRL)*, vol. 49(4), pp. 391–409, 2002.

[161] G. Casale, N. Mi, and E. Smirni, "Bound Analysis of Closed Queueing Networks with Workload Burstiness," in *SIGMETRICS'08, Annapolis, MD*, 2008.

[162] D. Heyman, A. Tabatabai, and T. Lakshman, "Statistical Analysis and Simulation Study of Video Teleconference Traffic in ATM Networks," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 2(1), pp. 49–58, 1992.

[163] A. Alheraish, S. Alshebeili, and T. Alamri, "A GACS Modeling Approach for MPEG Broadcast Video," *IEEE Trans. Broadcast.*, vol. 50(2), pp. 132–141, 2004.

[164] A. Papoulis and S. Pillai, *Probability, Random Variables, and Stochastic Processes, 4th Edition.* McGrawn-Hill, 2002.

[165] G. Box and M. Muller, "A Note on the Generation of Random Normal Deviates," *The Annals of Mathematical Statistics*, vol. 29(2), pp. 610–611, 1958.

[166] G. Box, G. Jenkins, and G. Reinsel, *Time Series Analysis - Forecasting and Control, 4th Edition.* Wiley, 2008.

[167] P. Barford and M. Crovella, "Generating Representative Web Workloads for Network and Server Performance Evaluation," in *SIGMETRICS'98, Madison, WI*, 1998.

[168] W. Willinger, V. Paxson, and M. Taqqu, "Self-Similarity and Heavy Tails: Structural Modeling of Network Traffic," in *A Practical Guide to Heavy Tails* (R. Adler, ed.), ch. 1, pp. 27–54, Birkhauser, 1998.

[169] P. Svoboda, "Traffic Flows," in *Video and Multimedia Transmissions over Cellular Networks: Analysis, Modelling and Optimization in Live 3G Mobile Networks* (M. Rupp, ed.), ch. 13, Wiley, 2009.

[170] P. Svoboda, W. Karner, and M. Rupp, "Traffic Analysis and Modeling for World of Warcraft a MMOG," in *ICC'07, Glasgow, Scotland*, 2007.

[171] W. Leland *et al.*, "On the self-similar nature of Ethernet traffic (extended version) ," *IEEE/ACM Trans. Netw.*, vol. 2(1), pp. 1–15, 1994.

[172] I. Norros, "On the Use of Farctional Brownian Motion in Theory of Connectionless Networks," *IEEE J. Sel. Areas Commun.*, vol. 13(6), pp. 953–962, 1995.

[173] P. Branch and G. Armitage, "Measuring the auto-correlation of server to client traffic in First Person Shooter games," in *ATNAC, Melbourne, Australia*, 2006.

[174] M. Dai, Y. Zhang, and D. Loguinov, "A Unified Traffic Model for MPEG-4 and H.264 Video Traces," *IEEE Trans. Multimedia*, vol. 11(5), pp. 1010–1023, 2009.

[175] M. Garrett and W. Willinger, "Analysis, Modeling and Generation of Self-Similar VBR Video Traffic," in *SIGCOMM'94, London, UK*, 1994.

[176] P. Abry *et al.*, "Revisiting an old friend: On the observability of the relation between Long Range Dependence and Heavy Tail," *Springer Telecommun. Sys.*, vol. 43, pp. 147–165, 2010.

[177] P. Brockwell and R. Davis, *Time Series: Theory and Methods, 2nd Edition.* Springer, 1991.

[178] F. Bause, P. Buchholz, and J. Kriege, "A Comparison of Markovian Arrival and ARMA/ARTA Processes for the Modeling of Correlated Input Processes," in *WSC'09, Austin, TX*, 2009.

[179] A. Dainotti *et al.*, "Internet traffic modeling by means of Hidden Markov Models," *Elsevier Computer Netw.*, vol. 52, pp. 2645–2662, 2008.

[180] G. Casale, E. Z. Zhang, and E. Smirni, "Trace data characterization and fitting for Markov modeling," *Elsevier Perform. Eval.*, vol. 67 (2), pp. 61–79, 2010.

[181] M. Menth, A. Binzenhöfer, and S. Mühleck, "Source Models for Speech Traffic Revisited," *IEEE/ACM Trans. Netw.*, vol. 17(4), pp. 1042–1051, 2009.

[182] P. Branch, A. Cricenti, and G. Armitage, "A Markov Model of Server to Client IP Traffic in First Person Shooter Games," in *ICC'08, Beijing, China*, 2008.

[183] B. Melamed, "An Overview of TES Processes and Modeling Methodology," in *Performance Evaluation of Computer and Communication Systems* (L. Donatiello, ed.), ch. 1, pp. 359–393, Springer, 1993.

[184] D. L. Jagerman and B. Melamed, "The Transition and Autocorrelation Structure of TES Processes," *Commun. Stat. Stochastic Models*, vol. 8(2), pp. 193–219, 1992.

[185] B. Liu and D. Munson, "Generation of a Random Sequence Having a Jointly Specified Marginal Distribution and Autocovariance," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 30(6), pp. 973–983, 1982.

[186] M. Grigoriu, *Applied NonGaussian Processes.* Prentice Hall, 1995.

[187] M. Cario and B. Nelson, "Numerical Methods for Fitting and Simulating Autoregressive-To-Anything Processes," *INFORMS J. Computing*, vol. 10(1), pp. 72–81, 1997.

[188] H. Helgason, V. Pipiras, and P. Abry, "Synthesis of multivariate stationary series with prescribed marginal distributions and covariance using circulant matrix embedding," *Elsevier Signal Process.*, vol. 91(8), pp. 1741–1758, 2011.

[189] W. Palma, *Long-Memory Time Series.* Wiley, 2007.

[190] J. Kriege and P. Buchholz, "Correlated phase-type distributed random numbers as input models for simulations," *Elsevier Perform. Eval.*, vol. 68(11), pp. 1247–1260, 2011.

[191] B. Biller and B. Nelson, "Modeling and Generating Multivariate Time-series Input Processes Using a Vector Autoregressive Technique," *ACM Trans. Model. Comput. Simul.*, vol. 13(3), pp. 211–237, 2003.

[192] X. Huang, Y. Zhou, and R. Zhang, "A Multiscale Model for MPEG-4 Varied Bit Rate Video Traffic," *IEEE Trans. Broadcast.*, vol. 50(3), pp. 323–334, 2004.

[193] C. Liew, C. Kodikara, and A. Kondoz, "MPEG-encoded Variable Bit Rate Video Traffic Modelling," *IEE Proc. Communications*, vol. 152(5), pp. 749–756, 2005.

[194] P. Borgnat, P. Abry, and P. Flandrin, "Using Surrogates And Optimal Tansport For Synthesis of Stationary Multivariate Series With Prescribed Covariance Function And Non-Gaussian Joint-Distribution," in *ICASSP'12, Kyoto, Japan*, 2012.

[195] C. Lawson and R. Hanson, *Solving Least Squares Problems.* Prentice-Hall, 1974.

[196] M. Matsumoto and T. Nishimura, "Mersenne Twister: A 623-dimensionally Equidistributed Uniform Pseudo-random Number Generator," *ACM Trans. Model. Comput. Simul.*, vol. 8(1), pp. 3–30, 1998.

[197] V. Paxson and S. Floyd, "Wide-Area Traffic: The Failure of Poisson Modeling," *IEEE/ACM Trans. Netw.*, vol. 3(3), pp. 226–244, 1995.

[198] J. Liu *et al.*, "Traffic Modeling Based on FARIMA Models," in *CCECE'99, Edmonton, Canada*, 1999.

[199] J. Nolan, *Stable Distributions - Models for Heavy Tailed Data.* Birkhauser, 2012.

[200] R. Riedi *et al.*, "A Multifractal Wavelet Model with Application to Network Traffic," *IEEE Trans. Inf. Theory*, vol. 45(3), pp. 992–1018, 1999.

[201] G. Samorodnitsky, "Long Range Dependence," *Encycl. Actuarial Science*, vol. 1, pp. 163–257, 2006.

[202] "The ACM/SIGCOMM Internet Traffic Archive." [Online]. Available: http://ita.ee.lbl.gov.

[203] "OpenArena, a FPS online game." [Online]. Available: http://openarena.ws.

[204] P. Seeling and M. Reisslein, "Video Transport Evaluation With H.264 Video Traces," *IEEE Commun. Surveys Tuts.*, vol. 14(4), pp. 1142–1165, 2012. [Online]. Available: trace.eas.asu.edu/publications/H264VidTraceTut.pdf.

[205] A. S. Alfa, "Matrix-Geometric Solution of Discrete Time MAP / PH / 1 Priority Queue," *Wiley Naval Research Logistics*, vol. 45(1), pp. 23–50, 1998.

[206] 3GPP, "TR 22.368, Service Requirements for Machine-Type Communications." [Online]. Available: www.3gpp.org.

[207] Y. Morioka, "LTE for Mobile Consumer Devices," in *ETSI Workshop on Machine to Machine Standardization*, 2011.

[208] M. Z. Shafiq *et al.*, "A First Look at Cellular Machine-to-Machine Trafffic – Large Scale Measurements and Characterization," in *SIGMETRICS/Performance'12, London, UK*, 2012.

[209] 3GPP, "TR 37.868, Study on RAN Improvements for Machine-type communications." [Online]. Available: www.3gpp.org.

[210] H. Heffes and D. M. Lucatoni, "A Markov Modulated Characterization of Packetized Voice and Data Traffic and Related Statistical Multiplexer Performance," *IEEE J. Sel. Areas Commun.*, vol. 4(6), pp. 856–868, 1986.

[211] R. C. D. Paiva, R. D. Vieira, and M. Säily, "Random access capacity evaluation with synchronized MTC users over wireless networks," in *VTC Spring '11, Brasilia, Brazil*, 2011.

[212] G. Wu *et al.*, "M2M: From Mobile to Embedded Internet," *IEEE Commun. Mag.*, vol. 49(4), pp. 36–43, 2011.

[213] D. Drajic *et al.*, "Impact of online games and M2M applications traffic on performance of HSPA radio access networks," in *esIoT'12, Palermo, Italy*, 2012.

[214] K. Zhou *et al.*, "Contention Based Access for Machine-Type Communications over LTE," in *VTC Spring '12, Yokohama, Japan*, 2012.

[215] R. Ratasuk, J. Tan, and A. Ghosh, "Coverage and Capacity Analysis for Machine Type Communications in LTE," in *VTC Spring '12, Yokohama, Japan*, 2012.

[216] S. Lien, K. Chen, and Y. Lin, "Toward Ubiquitous Massive Accesses in 3GPP Machine-to-Machine Communications," *IEEE Commun. Mag.*, vol. 49(4), pp. 66–74, 2011.

[217] Y. Jou *et al.*, "M2M over CDMA2000 1x Case Studies," in *WCNC'11, Cancun, Mexico*, 2011.

[218] Y. Zhang *et al.*, "Home M2M Networks: Architectures, Standards, and QoS Improvements," *IEEE Commun. Mag.*, vol. 49(4), pp. 44–52, 2011.

[219] M. Brand, N. Oliver, and A. Pentland, "Coupled hidden Markov models for complex action recognition," in *CVPR'97, San Juan, PR*, 1997.

[220] M. Brand, "Coupled hidden Markov models for modeling interacting processes," tech. rep., MIT Media Lab, 1997. [Online]. Available: www.media.mit.edu/~brand/papers/brand-chmm.ps.gz.

[221] N. Nikaein *et al.*, "Simple Traffic Modeling Framework for Machine Type Communication," in *ISWCS'13, Ilmenau, Germany*, 2013.

[222] R. Kwan, C. Leung, and J. Zhang, "Multiuser scheduling on the downlink of an LTE cellular system," *Rec. Lett. Commun.*, vol. 3, pp. 1–4, 2008.

[223] Z. Sun, C. Yin, and G. Yue, "Reduced-Complexity Proportional Fair Scheduling for OFDMA Systems," in *ICCCAS'06, Guilin, China*, 2006.

[224] S. Schwarz, C. Mehlführer, and M. Rupp, "Low Complexity Approximate Maximum Throughput Scheduling for LTE," in *ASILOMAR'10, Pacific Grove, CA*, 2010.

[225] 3GPP, TSG-RAN WG1#48, "R1-070674, LTE physical layer framework for performance verification," 2007. St. Louis, MI, [Online]. Available: www.3gpp.org/ftp/tsg_ran/WG1_RL1/TSGR1_48/Docs/R1-070674.zip.

[226] J. Evans and D. Everitt, "On the Teletraffic Capacity of CDMA Cellular Networks," *IEEE Trans. Veh. Technol.*, vol. 48, pp. 153–165, 1999.

[227] A. Klemm, C. Lindemann, and M. Lohmann, "Traffic Modeling and Characterization for UMTS Networks," in *GLOBECOM'01, San Antonio, TX*, 2001.

[228] A. Balachandran, G. Voelker, P. Bahl, and P. Rangan, "Characterizing User Behavior and Network Performance in a Public Wireless LAN," in *SIGMETRICS'02, Marina Del Rey, CA*, 2002.

[229] D. Willkomm, S. Machiraju, J. Bolot, and A. Wolisz, "Primary User Behavior in Cellular Networks and Implications for Dynamic Spectrum Access," *IEEE Commun. Mag.*, vol. 47, pp. 88–95, 2009.

[230] U. Paul *et al.*, "Understanding Traffic Dynamic in Cellular Data Networks," in *INFOCOM'11, Shanghai, China*, 2011.

[231] R. Keralapura *et al.*, "Profiling Users in a 3G Network Using Hourglass Co-Clustering," in *MobiCom'10, Chicago, IL*, 2010.

[232] M. Alvarez-Campana, E. Vazquez, and J. Vinyes, "Performance Modeling of Web Access over HSPA Networks," in *WMCNT'10, Moscow, Russia*, 2010.

[233] I. Aban, M. Meerschaert, and A. Panorska, "Parameter Estimation for the Truncated Pareto Distribution," *J. American Stat. Ass.*, vol. 101(473), pp. 270–277, 2006.

[234] W. Cleveland, D. Lin, and D. Sun, "IP Packet Generation: Statistical Models for TCP Start Times Based on Connection-rate Superposition," in *SIGMETRICS'00, Santa Clara, CA*, 2000.

[235] J. Cao, W. Cleveland, D. Lin, and D. Sun, "Internet Traffic Tends *Toward* Poisson and Independent as the Load Increases," in *Nonlinear Estimation and Classification*, pp. 83–109, Springer, 2002.

[236] C. Mehlführer *et al.*, "The Vienna LTE simulators — Enabling Reproducibility in Wireless Communications Research," *EURASIP JASP*, vol. 2011(29), pp. 1–14, 2011.

[237] J. Colom Ikuno, M. Wrulich, and M. Rupp, "System Level Simulation of LTE Networks," in *VTC'10, Taipei, Taiwan*, 2010.

[238] IETF, "RFC 791: Internet Protocol," 1981. [Online]. Available: www.ietf.org/rfc/rfc791.txt.

[239] J. Postel, "RFC 768, User Datagram Protocol," 1980. [Online]. Available: www.ietf.org/rfc/rfc793.txt.

[240] "Network Time Protocol." [Online]. Available: www.ntp.org.

[241] J. Ridoux and D. Veicht, "A Methodology for Clock Benchmarking," in *TridentCom'07, Lake Buena Vista, FL*, 2007.

[242] C. Hong, C. Lin, and M. Caesar, "Clockscalpel: Understanding Root Causes of Internet Clock Synchronization Inaccuracy," in *PAM'11, Atlanta, GA*, 2011.

[243] K. Sato and K. Asari, "Characteristics of Time Synchronization Response of NTP Clients on MS Windows OS and Linux OS," in *38th PTTI Meeting, Washington, DC*, 2006.

[244] "gpsd – a GPS service daemon." [Online]. Available: gpsd.berlios.de.

[245] Intel, "Interrupt Moderation Using Intel(R) GbE Controllers," tech. rep., Intel, 2007. [Online]. Available: www.intel.com/content/dam/doc/application-note/gbe-controllers-interrupt-moderation-appl-note.pdf.

[246] A. Disslbacher-Fink, "Hardware Based Timing Synchronization," Master's thesis, Vienna University of Technology, 2011. [Online]. Available: publik.tuwien.ac.at/files/PubDat_195758.pdf.

[247] J. Coleman, "Reducing Interrupt Latency Through the Use of Message Signaled Interrupts," tech. rep., Intel, 2009. [Online]. Available: www.intel.com/content/dam/www/public/us/en/documents/white-papers/msg-signaled-interrupts-paper.pdf.

[248] L. Isserlis, "On a Formula for the Product-Moment Coefficient of Any Order of a Normal Frequency Distribution in Any Number of Variables," *Biometrika*, vol. 12(1), pp. 134–139, 1918.

[249] "NIST, Digital Library of Mathematical Functions, Sec. 26.4." [Online]. Available: dlmf.nist.gov/26.4.

[250] "OEIS, Sequence A001147, Double factorial numbers." [Online]. Available: http://oeis.org/A001147.

[251] W. Gong *et al.*, "Self-similarity and long range dependence on the internet: a second look at the evidence, origins and implications," *Elsevier Computer Netw.*, vol. 48, pp. 377–399, 2005.

[252] A. T. Andersen and B. F. Nielsen, "A Markovian Approach for Modeling Packet Traffic with Long-Range Dependence," *IEEE J. Sel. Areas Commun.*, vol. 16(5), pp. 719–732, 1998.

[253] R. Fletcher, *Practical methods of optimization (2nd Ed.)*. Wiley, 1987.

[254] K. P. Tsoukatos and A. M. Makowski, "Power-law vs. Esponential Queueing in a Network Traffic Model," *Elsevier Perform. Eval.*, vol. 65, pp. 32–50, 2008.

[255] Y. Chakhchoukh, "A New Robust Estimation Method for ARMA Models," *IEEE Trans. Signal Process.*, vol. 58(7), pp. 3512–3522, 2010.

[256] G. Beylkin and L. Monzon, "On Approximation of Functions by Exponential Sums," *Elsevier Appl. Comput. Harmon. Anal.*, vol. 19, pp. 17–48, 2005.

[257] S. Li and C. Hwang, "On the Convergence of Traffic Measurement and Queueing Analysis: A Statistical-Matching and Queueing (SMAQ) Tool," *IEEE/ACM Trans. Netw.*, vol. 5(1), pp. 95–110, 1997.

[258] P. Salvador and R. Valadas, "Multiscale Fitting Procedure Using Markov Modulated Poisson Processes," *Telecommun. Systems*, vol. 23(1), pp. 123–148, 2003.

[259] R. A. Khayari, R. Sadre, and B. R. Haverkort, "Fitting world-wide web request traces with the EM-algorithm," *Elsevier Perform. Eval.*, vol. 52, pp. 175–191, 2003.

[260] M. Grossglauser and J. Bolot, "On the Relevance of Long-Range Dependence in Network Traffic," *IEEE/ACM Trans. Netw.*, vol. 7(5), pp. 629–640, 1999.

[261] G.Casale, E.Z.Zhang, and E.Smirni., "KPC-Toolbox: Best Recipes for Automatic Trace Fitting Using Markovian Arrival Processes," *Elsevier Perform. Eval.*, vol. 67(9), pp. 873–896, 2010.

Abbreviations

3G	3rd Generation
3GPP	3rd Generation Partnership Project
ACF	Auto-correlation Function
AR	Auto-Regressive
ARFIMA	Auto-Regressive Fractionally Integrated Moving-Average
ARMA	Auto-Regressive Moving-Average
ARTA	Auto-Regressive To Anything
ASN.1	Abstract Syntax Notation One
ATM	Asynchronous Transfer Mode
CBR	Constant Bit-Rate
CCDF	Complementary Cumulative Distribution Function
CDF	Cumulative Distribution Function
CMMPP	Coupled Markov Modulated Poisson Processes
DNS	Domain Name System
DUT	Device Under Test
ECCDF	Empirical Complementary Cumulative Distribution Function
ECDF	Empirical Cumulative Distribution Function
FIFO	First In – First Out
FPGA	Field Programmable Gate Array
FTP	File Transfer Protocol
GACS	Gaussian Auto-Regressive and Chi-Squared
GGSN	Gateway GPRS Support Node
GMT	Greenwich Mean Time
GOP	Group of Picture
GPS	Global Positioning System
GSM	Global System for Mobile Communications
HSDPA	High Speed Downlink Packet Access
HSPA	High Speed Packet Access
HSUPA	High Speed Uplink Packet Access
HTC	Human-type Communication
HTTP	Hypertext Transfer Protocol
IAT	Inter Packet-Arrival Time
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
i.i.d.	independent identically distributed
IP	Internet Protocol
IPFIX	IP Flow Information Export

IPv4	Internet Protocol Version 4
IPPM	Internet Protocol Performance Metrics
ISO	International Organization for Standardization
ITU	International Telecommunication Union
LIFO	Last In – First Out
LRD	Long Range Dependence
LTE	Long Term Evolution
LTI	Linear Time Invariant
M2M	Machine-to-Machine Communication
MA	Moving-Average
MAP	Markovian Arrival Process
MMPP	Markov Modulated Poisson Process
ML	Maximum Likelihood
MTC	Machine-type Communication
MTU	Maximum Transmission Unit
NLS	Non-linear Least Squares
NodeB	Base Station
NTP	Network Time Protocol
OSI	Open System Interconnection
OWD	One-Way Delay
PASTA	Poisson Arrivals See Time Averages
PDF	Probability Density Function
PDU	Protocol Data Unit
PMF	Probability Mass Function
PNPN	Priority Service
PPP	Point-to-point Protocol
PPS	Pulse Per Second
PS	Packet Size
PSD	Power Spectral Density
PTP	Precision Time Protocol
QoS	Quality of Service
REF	Reference
RF	Radio Frequency
RFC	Request For Comment
RGCH	Relative Grant Channel
RLC	Radio Link Control
RNC	Radio Network Controller
RTP	Real Time Protocol
RTT	Round-Trip Time
SDU	Service Data Unit
SIRO	Service In Random Order
SSH	Secure Shell
SyncE	Synchronous Ethernet
TAI	International Atomic Time
TARMA	Transformed Auto-Regressive Moving-Average
TCP	Transmission Control Protocol
TLS/SSL	Transport Layer Security
TTI	Transmission Time Interval
UDP	User Datagram Protocol
UE	User Equipment

UMTS	Universal Mobile Telecommunication System
U-RNTI	UTRAN Radio Network Temporary Identifier
UTC	Coordinated Universal Time
UTRAN	UMTS Terrestrial Radio Access Network
VBR	Variable Bit-Rate
VoIP	Voice over IP
WCDMA	Wide-band Code Division Multiple Access
WLAN	Wireless Local Area Network
XCF	Cross-correlation Function
XML	Extensible Markup Language

Symbols

Symbol	Description	Unit
D_n	n-th data-unit	
n, m	index of data-unit	
N	number of data-units	
X, Y, A, B, C	interfaces between links and nodes	
$\delta_{XY}[n]$	delay of D_n between interfaces X and Y	s
$\Delta_{\max}$	maximum delay	s
$L_{\max}$	maximum packet loss ratio	
$t_{X,n,i}$	i-th timestamp of D_n at interface X	s
i, ι, υ	index of timestamp	
I	number of timestamps per data-unit	
$A(t)$	packet arrival process	1/s
$D(t)$	packet departure process	1/s
$N(t)$	packets in the system	
δ_p	processing delay	s
δ_q	queueing delay	s
δ_t	transmission delay	s
δ_{prop}	propagation delay	s
A	arrival process	
S	service time distribution	
c	number of servers	
k	buffer length	
N	calling population	
D	queueing discipline	
λ	average arrival rate	1/s
μ	average service rate	1/s
U	queue utilization	
$C_{\max}$	link capacity (max. throughput)	B/s
$R_A(t)$	average arrival rate until t	1/s
$R_D(t)$	average departure rate until t	1/s
$\delta_r[n]$	response time, nth packet	s
$\overline{N}$	average number of customers in a syst.	
$\overline{\delta_r}$	average response time	s
Θ	Parameter set (probing traffic)	
θ	sample of Θ	

Ψ	Parameter set (fast fluctuations)	
ψ	sample of Ψ	
Φ	Parameter set (slow effects)	
ϕ	sample of Φ	
$\pi[n]$	packet size of nth packet	B
$\tau[n]$	IAT btw. $n-1$st and nth packet	s
$r[n]$	instantaneous rate	B/s
$C_{\max}$	link capacity	B/s
U_p	maximum utilization due to probing	
P_{NE}	prob. of influences from history, cf. Eq. (B.1)	
$Z[n]$	rand. process, arbitrary distribution	
$Y[n]$	rand. process, normal distribution	
$X[n]$	normal i.i.d. random process	
$Q^{-1}(\cdot)$	inverse Q-function	
φ	weigthing factor	
ρ	(auto)-correlation coefficient	
$\rho_{\tau,\pi}$	cross-correlation coefficient	
t_c	time instant according to UTC	s
$T(t_c)$	time reported by a clock at t_c	
$\Delta T(t_c)$	offset of a clock at t_c	
$\Delta f(t_c)$	skew of a clock at t_c	Hz
$D(t_c)$	drift of a clock at t_c	Hz/s
$\Delta T_{\max}$	maximum clock offset during the measurement	s
T_Θ	temporal horizon (traffic pattern)	s
T_Ψ	independence distance (fast fluctuations)	s
T_Φ	short period (slow effects)	s
R	global data rate	B/s
R_c	constant global data rate	B/s
$\pi_{\min}$, $\pi_{\max}$	bounds for the Packet Size (PS)	B
$\tau_{\min}$, $\tau_{\max}$	bounds for the Inter Packet-Arrival Time (IAT)	s
$R_{\min}$, $R_{\max}$	bounds for the data rate	B/s
ϵ	small constant	
η	number of CDF samples per decade	
$D_2(\cdot,\cdot)$	CDF distance	
$F(\delta), G(\delta)$	delay CDFs	
$\mathbf{f}, \mathbf{g}$	vectorized delay CDFs	
K	number of clusters	
F_{VBR}	delay CDF, variable bit rate probing	
F_{CBR}	delay CDF, constant bit rate probing	
$\mathbf{s}_X[i]$	ith vectorized CDF of session type X	
$\overline{\mathbf{s}_X}$	average vectorized CDF of session type X	
$d_X[i]$	distance of ith vectorized CDF	
$t_{10\%}$	time after which ρ<10%	s
$P_{R,U}$	retransmission probability uplink	
$P_{R,D}$	retransmission probability downlink	
t_{Px}	timestamp at probe x	s
$\delta_{x,U}$	delay over the component x in uplink	s

$\delta_{x,D}$	delay over the component x in downlink	s
$D_{\max}$	maximum CDF distance	
$Z_i[n], Y_i[n], X_i[n], W_j[n]$	random process	
I, J	number of processes	
n	time index	
m	lag index	
$\mathfrak{p}_Y(\cdot)$	polynomial transf. of $Y[n]$	
α_p, β_p	polynomial coefficients	
P, Q	polynomial order	
$\mathfrak{p}_{\rho,ii}(\cdot)$	polyn. transf. of $\rho_{Y_iY_i}[m]$	
$h[m]$	impulse response	
$\phi(B), \theta(B)$	AR and MA polynomials	
B	backshift operator	
$\rho_{ZZ}[m]$, $\rho_{Z_iZ_i}[m]$	normalized ACF of $Z[n]$	
$\gamma_{ZZ}[m]$	unnormalized ACF	
$\rho_{Z_iZ_l}[m]$	normalized XCF of $Z_i[n]$, $Z_l[n]$	
$\mathbf{G}[m]$	weighting matrix	
$g_{ij}[m]$	element of $\mathbf{G}[m]$	
$\mathbf{G}(B)$	weighting matrix, B-domain	
$g_{ij}(B)$	element of $\mathbf{G}(B)$	
$\mathbf{\Gamma}[m]$	XCF matrix	
$\mathbf{\Gamma}(B)$	XCF matrix, B-domain	
$\mathsf{m}_Z^{(l)}$	l-th moment of $Z[n]$	
$F_Z(z), F_Y(y)$	CDF of $Z[n]$, $Y[n]$	
$f_Z(z), f_Y(y)$	PDF of $Z[n]$, $Y[n]$	
$\Omega_{Z,k}, \Omega_{Y,k}$	ω_k-quantile of $Z[n]$, $Y[n]$	
N	number of M2M devices	
$f_T(t)$	distribution of arrivals	1/s
T	simulation period	s
Δt	time slot duration	s
k	time slot index	
K	number of time slots	
$\lambda(k)$	arrival rate in time slot k	1/s
$\mathbf{P}$	state transition matrix	
π	state probability vector	
λ_i	arrival rate in state i	1/s
λ_g	global arrival rate	1/s
$\mathbf{P}_C$	coordinated state trans. matrix	
$\mathbf{P}_U$	uncoordinated state trans. matrix	
ζ_n	constant weight	
$\Xi(t)$	background process (master)	
$\xi[k]$	sample of $\Xi(t)$ at $k{\cdot}\Delta t$	
$\xi_n[k]$	weight for the nth device	
$\mathbf{P}_n[k]$	state trans. matrix for device n at time slot k	
$s_n[k]$	state of device n at time slot k	
κ	constant weight	
Σ	set of simulation parameters	

$D_C(\cdot,\cdot)$	Cramer's distance (rand. var.)	
N_u	number of active users	
λ_n	mean number of active users	1/s
$\overline{R}_c$	mean throughput per cell	bit/s
L_s	session duration	s
α_L	Pareto index of L_s	
A_s	session arrival rate	1/s
λ_A	mean session arrival rate	1/s
$\overline{R}_s$	mean session throughput	bit/s
α_R	Pareto index of $\overline{R}_s$	

Index