Julius-Maximilians-Universität Würzburg
Institut für Informatik
Lehrstuhl für Kommunikationsnetze
Prof. Dr.-Ing. P. Tran-Gia

Modeling and Evaluation of Multi-Stakeholder Scenarios in Communication Networks

Christian Schwartz

Würzburger Beiträge zur
Leistungsbewertung Verteilter Systeme

Bericht 1/16

Würzburger Beiträge zur Leistungsbewertung Verteilter Systeme

Herausgeber

Prof. Dr.-Ing. P. Tran-Gia
Universität Würzburg
Institut für Informatik
Lehrstuhl für Kommunikationsnetze
Am Hubland
D-97074 Würzburg

Tel.: +49-931-31-86630
Fax.: +49-931-31-86632
email: trangia@informatik.uni-wuerzburg.de

Satz

Reproduktionsfähige Vorlage des Autors.
Gesetzt in LaTeX Linux Libertine 10pt.

ISSN 1432-8801

Modeling and Evaluation of Multi-Stakeholder Scenarios in Communication Networks

Dissertation zur Erlangung des
naturwissenschaftlichen Doktorgrades
der Julius–Maximilians–Universität Würzburg

vorgelegt von

Christian Schwartz

aus

Fulda

Würzburg 2016

Eingereicht am: 22.01.2016
bei der Fakultät für Mathematik und Informatik
1. Gutachter: Prof. Dr.-Ing. Phuoc Tran-Gia
2. Gutachter: Prof. Dr. Franco Davoli
Tag der mündlichen Prüfung: 20.04.2016

Danksagung

Prof. Dr. Tran-Gia hat die Rahmenbedingungen geschaffen, unter denen diese Dissertation entstehen konnte. Unter seiner Leitung entstand am Lehrstuhl eine Atmosphäre, in der sowohl wissenschaftlicher Diskurs als auch ein freundliches Miteinander stattfinden konnte. Er hat mir die Gelegenheit gegeben an wissenschaftlichen und industriellen Kooperationen im In- und Ausland teilzunehmen, die mich und den Inhalt dieser Dissertation maßgeblich geprägt haben. Er selbst stand immer für fachliche Diskussionen zur Verfügung und war, insbesondere für die theoretischeren Aspekte dieser Arbeit, meine Anlaufstelle. Ich danke ihm außerdem für das Vertrauen in mich, das mir diese Arbeit erst ermöglicht hat.

Ich danke natürlich auch meinem Zweitgutachter, Prof. Dr. Davoli, für die Bereitschaft meine Dissertation zu begutachten.

Den Mitgliedern der Prüfungskommission meiner Disputation, Prof. Dr. Kolla und Prof. Dr. Hotho, danke ich für die Flexibilität bei der Terminfindung und der freundlichen Atmosphäre während der Prüfung.

Dank gebührt natürlich auch Alison Wichmann für die Unterstützung bei der Bewältigung der zahlreichen administrativen Hürden im wissenschaftlichen und projektbezogenen Alltag.

Mit Prof. Dr. Wehnes konnte ich auch im Rahmen eines Industrieprojektes zusammenarbeiten. Aus seiner Erfahrung im Bereich Projektmanagement konnte ich persönlich viel Lernen. Hierfür danke ich ihm.

Meine Kollegen Matthias Hirth, Anh Nguyen-Ngoc, Nicholas Gray, Christopher Metter, Michael Seufert, Valentin Burger, Kathrin Borchert, Lam Dinh-Xuan, Dr. Florian Wamser, Steffen Gebert, Stanislav Lange und Dr. Thomas Zin-

ner sind im Laufe der Arbeit am Lehrstuhl und insbesondere im Kaffeeraum zu Freunden geworden, für die ich in den letzten neun Monaten meiner Arbeitszeit die langen Zugfahrten zwischen Darmstadt und Würzburg gerne in Kauf genommen habe.

Auch außerhalb der Arbeitszeit am Lehrstuhl haben mich einige meiner Kollegen begleitet. In vielen virtuellen Expeditionen haben mich Matthias Hirth und Valentin Burger durch Siege und Niederlage begleitet. Kathrin Borchert und Nicholas Gray waren immer gerne zu abendlichen Spielerunden bereit. Mein Mitstreiter im Bereich DevOps, Steffen Gebert, hat mit mir Workshops zum Thema DevOps besucht, die ich zu den Highlights der von mir besuchten Konferenzen zähle. Euch allen vielen Dank für eure Freundschaft und die schönen Stunden auch außerhalb des Lehrstuhls.

Bedanken möchte ich mich auch bei meinen Gruppenleitern Dr. Rastin Pries, Prof. Dr. Tobias Hoßfeld und Dr. Florian Wamser. Während meiner wissenschaftlichen Karriere standen sie mir stets mit Rat und Tat zur Seite, waren immer gerne bereit neue und interessante Fragestellungen zu diskutieren, haben mir aber auch, gerade gegen Ende meiner Zeit am Lehrstuhl, die notwendigen Freiheiten gegeben für mich eigene Themengebiete zu definieren.

Zahlreiche Studenten haben mich während meiner Zeit am Lehrstuhl unterstützt, sei es bei der Betreuung von Vorlesungen oder der Durchführung von Industrieprojekten. Auch ihnen und den Studenten, mit denen ich im Rahmen von Seminar-, Praktikums- oder Abschlussarbeiten zusammenarbeiten konnte, gilt natürlich mein Dank.

Mit Dr. Florian Metzger konnte ich nur kurze Zeit am physikalisch selben Ort arbeiten. Trotzdem war er immer gerne bereit Themen aus allen Bereichen der Informatik zu diskutieren und darüber hinaus auch privat stets ein guter Freund.

Meine ehemalige Kollegen Dr. Michael Jarschel, Dr. Frank Lehrieder, Dr. Dominik Klein, Prof. Dr. Barbara Stähle, Prof. Dr. Dirk Stähle, Dr. Daniel Schlosser und Dr. Michael Duelli haben meinen wissenschaftlichen Arbeitsstil nicht unwesentlich, und natürlich nur positiv, beeinflusst.

Ich danke meinen Eltern, Sylvia und Bernd Schwartz, denn Sie haben mein

Interesse an der Informatik schon während meiner Schulzeit erkannt und gefördert. Sie haben mir mein Studium ermöglicht und damit wesentlich dazu beigetragen, dass ich meine Promotion erfolgreich durchführen konnte.

Nicht zuletzt danke ich meiner Frau, Prof. Dr. Alexandra Schwartz, und unserem Sohn Gabriel. Insbesondere in den letzten Monaten der Erstellung dieser Arbeit bekamen sie mich, durch die langen Fahrzeiten nur viel zu selten zu Gesicht. Trotzdem haben sie mich unterstützt und mir die Kraft gegeben, diese Arbeit zu einem guten Abschluss zu bringen. Danke.

Contents

1 Introduction

Today's Internet is no longer only controlled by a single stakeholder, e.g. a standard body or a telecommunications company. Rather, the interests of a multitude of stakeholders, e.g. application developers, hardware vendors, cloud operators, and network operators, collide during the development and operation of applications in the Internet. Each of these stakeholders considers different Key Performance Indicators (KPIs) to be important and attempts to optimise scenarios in its favour. This results in different, often opposing views and can cause problems for the complete network ecosystem.

One example of such a scenario are *Signalling Storms* [23] in the mobile Internet, with one of the largest occurring in Japan in 2012[1] due to the release and high popularity of a free instant messaging application. The network traffic generated by the application caused a high number of connections to the Internet being established and terminated. This resulted in a similarly high number of signalling messages in the mobile network, causing overload and a loss of service for 2.5 million users over 4 hours. While the network operator suffers the largest impact of this signalling overload, it does not control the application. Thus, the network operator can not change the application traffic characteristics to generate less network signalling traffic. The stakeholders who could prevent, or at least reduce, such behaviour, i.e. application developers or hardware vendors, have no direct benefit from modifying their products in such a way. This results in a clash of interests which negatively impacts the network performance for all participants.

[1] `https://www.techinasia.com/docomo-outage`, Accessed: November, 21st 2015

The goal of this monograph is to provide an overview over the complex structures of stakeholder relationships in today's Internet applications in mobile networks. To this end, we study different scenarios where such interests clash and suggest methods where tradeoffs can be optimised for all participants. If such an optimisation is not possible or attempts at it might lead to adverse effects, we discuss the reasons.

In the remainder of this chapter we first introduce the stakeholders considered in this work in Section 1.1. Then, in Section 1.2, we provide an overview over the scientific contributions of this monograph with respect to the stakeholders interest. Finally, Section 1.3 provides an outline of this monograph.

1.1 Scope of Considered Stakeholders

In this section we introduce the stakeholders considered in the remainder of this monograph and show their interactions in Figure 1.1.

First, we consider the *network operator*. The network operator owns, manages and operates a mobile network. By manipulating network configuration, the operator can influence the connection state of the User Equipment (UE), resulting in changes to the power drain, i.e. battery life, of the UE produced by the hardware vendor and reduced signalling load in the components of its mobile network.

The *application provider* develops and deploys applications and is interested in increasing the Quality of Experience (QoE) for the user, thus attracting a large user base. An additional considered KPIs for the application provider is cost, for example incurred due to use of compute or network resources in Infrastructure as a Service (IaaS) or Platform as a Service (PaaS) scenarios. Design and configuration of applications have an impact on traffic patterns which result in signalling traffic in the network of the mobile network operator and also influence the power drain of the UE of the hardware vendor.

These UEs are developed and sold by *hardware vendors*. While they theoretically implement standards proposed by the 3rd Generation Partnership Project

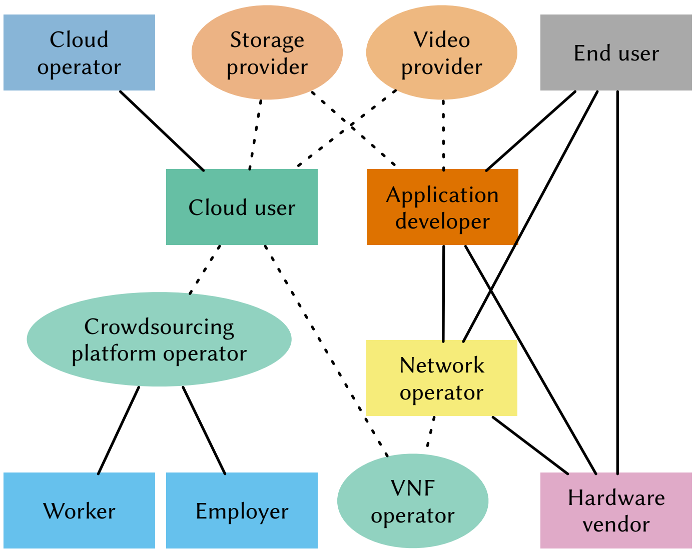


Figure 1.1: Stakeholder interactions considered in this monograph. Solid lines show stakeholder interactions, dotted lines show is-a relationships.

(3GPP) in order to establish connectivity with mobile networks, in reality vendors can deviate from this standard to increase their own KPIs. One example of such a deviation from a standard are proprietary fast dormancy mechanisms [24] implemented by some hardware vendors. These algorithms reduce power drain by disconnecting the UE earlier from the network to extend battery life and increase end user QoE. However, this has the consequence of increased signalling in the operator's network and can lead to increased web page loading times, i.e. decreased QoE from the end user's point of view.

End users employ UEs to execute applications in the system of the network operator. They are usually interested in increasing their QoE, i.e. by increasing the battery life of their UE or increasing satisfaction during video playback over the network.

Cloud operators provide services, i.e. compute, storage, or network resources, to cloud users according to specified Service Level Agreements (SLAs) for monetary compensation. They attempt to reduce the costs of operating infrastructure, e.g. by reducing power drain to increase revenue while still satisfying the SLA negotiated with the cloud users. Resources provided by cloud operators are purchased by *cloud users*. Further, they attempt to provide the best possible service to their own customers while reducing the number of required resources provided by the cloud operator in order to reduce cost of operation. In this monograph we consider two exemplary cloud users, which will be described in the following:

A *Virtualised Network Function (VNF) operator* uses virtualised resources obtained from a cloud operator to provide virtualised network services to other stakeholders. In the example considered in this monograph, the VNF operator uses cloud compute resources to provide a Gateway GPRS Support Node (GGSN) to a mobile network operator. The KPIs of the VNF operator are to satisfy the SLA with the mobile network operator and to reduce the use of compute resources of the cloud operator.

The second cloud user considered is the *crowdsourcing platform operator* who uses cloud resources in order to provide a crowdsourcing platform. The crowdsourcing platform operator in turn has to consider the requirements of its two main stakeholders: The *crowdsourcing employer* requires a set of microtasks to be completed in a short time to use the generated results in future business processes. *Crowdsourcing workers* complete microtasks for a specified amount of money. They are interested in completing as many tasks as possible and reduce their idle time, thus increasing their income.

The interactions of these stakeholders result in complex interactions, which are studied in this monograph. The next section introduces the considered interactions and provides an overview over the scientific contributions provided by this monograph.

1.2 Scientific Contribution

This monograph studies the interactions between different stakeholders in three, partially overlapping, scenarios in order to provide an overview of today's interlocking network and application ecosystem.

In Figure 1.2 we classify the areas of research as well as scientific methods used in relation to the chapters of this monograph. The x-axis shows the impacted areas of research, i.e. topics related to the mobile network, the application domain or cloud technologies. The y-axis details the applied scientific method. In the theoretical area methods from queueing theory, mean value analysis and the analysis of random variables are used. Measurements were performed using testbeds and custom software tools. Simulation studies, performed using Discrete Event Simulation (DES), and created analysis tools are summarised in the practical area.

Annotations are used to highlight scientific publication whose content contributes to the respective chapters.

The first contribution of this monograph is a discussion of the impact of mobile application traffic on mobile communication networks, especially considering the current network configuration. We study the impact of different traffic types, both from real-world applications and synthetic traffic distributions, and

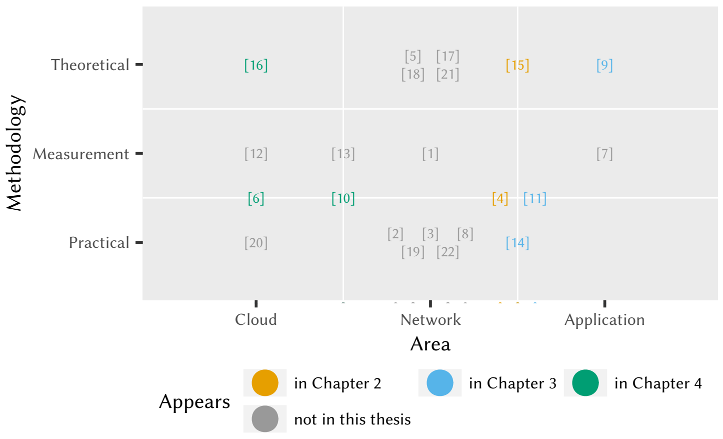


Figure 1.2: Contribution of this work as a classification of the research studies conducted by the author.

investigate the potential of network parameter optimisation as a means to reduce signalling traffic.

As a second contribution, we provide models for two popular applications, i.e. Video Streaming and Cloud File Synchronisation, thus enabling the study of the impact of different mechanisms implemented in these applications. We show that the streaming mechanism allows the most flexible configuration and can provide Pareto-optimal results for all pairs of metrics. However, further study shows that in fact no Pareto-optimal value exists which satisfies the KPIs of all participating stakeholders. Furthermore, we provide parametrisable QoE models for different user groups. For the Cloud File Synchronisation we compare different upload scheduling algorithms and find that, both the size based algorithm as well as the time based algorithm, can be used to specify a tradeoff between the different considered KPIs.

As a third contribution, we discuss the impact of resource dimensioning and management schemes in cloud environments. To this end, we study the performance of a power conservation mechanism for cloud environments using a queueing model. We derive guidelines for selecting Pareto-optimal results regarding both the waiting time before a job can begin processing and power drain of the cloud. Furthermore, we discuss a mechanism to reduce cost for cloud users by disabling compute instances while still allowing configurable SLAs and evaluate this mechanism using a queueing simulation. Finally, we present a mechanism to dimension worker numbers in a human-cloud scenario which can be used to ensure satisfaction of the key stakeholders of a crowdsourcing platform operator.

1.3 Outline of Thesis

In Chapter 2 we study the impact of mobile network configuration settings on participating stakeholders. First, we present an algorithm to infer power drain and signalling messages caused by a given application traffic. Then, we perform application traffic measurements and discuss general traffic characteristics before applying the introduced algorithm to the measurements. Finally, we generalise our results by introducing a theoretical model for power drain and signalling messages for arbitrary independent and identically distributed (iid) traffic distributions.

Chapter 3 focusses on the impact of applications design and choice of algorithm by the application developers on the other stakeholders. To this end, we study two prominent applications in today's Internet: Video Streaming and Cloud File Sychronisation. For Video Streaming, we study the impact of different video transmission mechanisms and parameter configurations on energy consumption of the UE, signalling in the mobile network and resource consumption at the application developer using a DES. Furthermore, we provide a queueing model for video streaming algorithms and derive a parametrised QoE model. To address the second scenario, we discuss different scheduling algorithms for Cloud File Synchronisation services. In order to accomplish this, we use data obtained from large scale, testbed based [25] measurements, implement a simulation model, and investigate relevant KPIs.

In Chapter 4 we study resource allocation strategies in the cloud and evaluate which management decisions of the cloud platform operators impact the other stakeholders. First, we consider an energy saving scheme where a cloud operator scales the number of available servers according to the available load. To evaluate this scenario, we introduce a queueing model and carry out a performance evaluation to study optimal parameter settings. Then, we consider the role of a cloud user renting virtual machines in the cloud to provide a service to users on the example of a virtualised network operator. We analyse traffic characteristics and use them as input for a simulation model of a virtualised GGSN. Combining these results, we evaluate the impact of different virtual server configurations and scaling strategies. Finally, we consider resource allocation in human-clouds. Based on data obtained from a commercial crowdsourcing provider, we extract characteristic distributions and apply them as input to both an analytic queueing model as well as a simulation model. Further, we derive dimensioning guidelines for the crowdsourcing provider.

In Chapter 5 we provide a summary of the major contributions of this work and suggest future potential research directions.

2 Impact of Application Traffic on Mobile Infrastructure

This chapter considers interactions between multiple stakeholders in future mobile networks and studies the resulting tradeoffs. These tradeoffs have only appeared recently with the advent of smartphones. With traditional cell phones, traffic in networks was largely dominated by voice traffic and to a smaller degree by text and signalling messages. The introduction of smartphones resulted in an increased amount of applications, developed by a decentralised developer community. With the application ecosystem no longer being under control of network operators, as in the case of voice or text messages, or hardware vendors, as with the rudimentary bundled applications of early feature phones, new types of network traffic occurred and replaced voice and signalling as the main traffic type. Furthermore, the amount of traffic is no longer the only performance indicator for the network operator, and the network operator is no longer the only stakeholder involved in the mobile network ecosystem.

Each of the stakeholders shown in Figure 2.1 is interested in optimising the network, device, or application performance in order to improve relevant Key Performance Indicators (KPIs). To this end, each of the stakeholders can manipulate the parts of the network that it controls. The *network operator* can change network configuration parameters in order to reduce *signalling* in the network. Hardware *vendors* can configure smartphones in such a way that data connections are terminated as soon as possible, decreasing *power drain. Application developers* can decrease polling intervals in their application layer protocols in order to increase *Quality of Experience (QoE).* However, the parameters

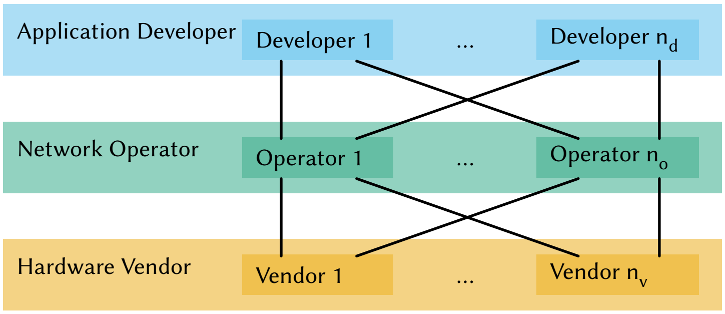


Figure 2.1: Stakeholders investigated in the network scenarios.

the stakeholders can influence in order to optimise the KPI of their individual concerns, also influence the complete network and thus all other KPIs, possible to the detriment of the other stakeholders. When a stakeholder attempts to optimise KPIs of interest, the consequences for other stakeholders have to be considered, as they could consider optimising their respective KPIs in turn, resulting in a net loss for all participating stakeholders. Thus, a tradeoff between all considered KPIs is required in order to satisfy the participating stakeholders.

Current best practices result in each participant optimising the respective KPIs individually, without addressing the needs of the other stakeholders [26, 27].

The contribution of this chapter is threefold:

a) We provide an algorithm to infer metrics for the relevant KPIs for the stakeholders from traffic traces, and evaluate exemplary traces for a set of popular applications.

b) We develop an analytical model in order to analyse theoretical and empirical application traffic distributions and derive KPIs for the stakeholders.

c) We study the impact of network timer optimisation, a practice where network operators modify network parameters in order to optimise signalling unilaterally, and show the impact for other stakeholders and highlight potential consequences for the network operator.

The content of this chapter is taken from [4, 15]. Its remainder is structured as follows. First, we give a background of mobile networks and survey related work in Section 2.1. In Section 2.2, we perform application traffic measurements and investigate the impact of application traffic on User Equipment (UE) power drain, network signalling and web QoE for a selected set of applications. Then, we generalise our results by introducing an analytical model in order to derive metrics for the state transition frequency and power drain from arbitrary traffic distributions in Section 2.3. Finally, we conclude this chapter with lessons learned in Section 2.4.

2.1 Background and Related Work

This section discusses the technical background relevant to the remainder of this chapter. First, in Section 2.1.1, we introduce the Universal Mobile Telecommunications System (UMTS) mobile communication standard, and the Radio Resource Control (RRC) protocol. Then, we discuss existing approaches to measure RRC protocol transactions and optimise the signalling load generated by RRC messages in Section 2.1.2. Finally, Section 2.1.3 tackles smartphone power drain and QoE, two metrics influenced by the configuration of the RRC protocol.

2.1.1 UMTS Networks and RRC Protocol

A Third Generation (3G) UMTS mobile network consists of three main components, which are depicted in Figure 2.2: the UE, the Radio Access Network (RAN), and the Core Network (CN). The RAN is used to establish connectivity between the UE and the CN, which in turn can establish connectivity to the Internet if required.

UEs are devices used by end users, i.e. smartphones, tablets or data card enabled notebooks, but can also include Machine to Machine (M2M) devices. The

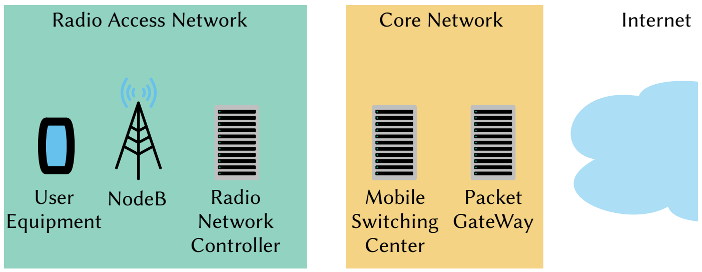


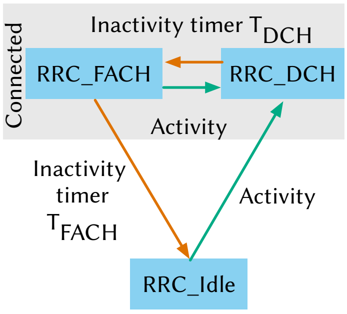
Figure 2.2: Overview of a 3G mobile network.

RAN is, amongst other tasks, responsible for RRC, packet scheduling and handover control. It includes the NodeB and the Radio Network Controller (RNC). The CN provides the backbone network of the UMTS network and provides connectivity to the Internet and the Public Switched Telephone Network (PSTN). Furthermore, the CN provides billing, authentication, and location management functionalities.

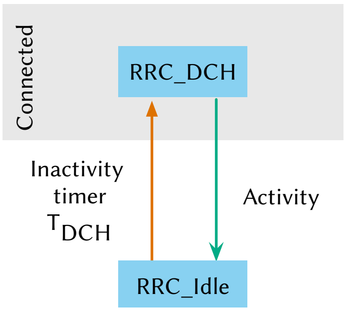
RRC Protocol In UMTS networks, the radio resources in the RAN between base station and UE are controlled and managed by the RRC protocol [28]. The protocol is responsible for control-band signalling between the UEs and the RAN. It is used to establish, maintain, and tear down connections between a UE and the RAN. Furthermore, the RRC protocol performs broadcasting of network information, Quality of Service (QoS) control, and reporting and cell selection management, which are out of scope for this text. The protocol is divided into different parts: services for upper layers, communication with lower layers, protocol states, RRC procedures, and error control. In particular, the RRC protocol also participates in the co-ordination of other resource management operations, channel measurements, and handovers. All RRC procedures rely on protocol states. The states are defined per UE and for the connection between the UE and the NodeB. Depending on the RRC state and network activity, originating or targeted at the UE, protocol actions can be triggered and transitions to other RRC states may occur. Typically there are five RRC states characterising a connection between UE and NodeB: RRC_Idle, URA_PCH, CELL_PCH, RRC_DCH, and RRC_FACH. Whether a specific RRC state is used in a specific mobile network depends on the configuration of the network by the provider. In the following we concentrate on the most commonly observed RRC states [29]: idle mode (RRC_Idle), Dedicated CHannel (CELL_DCH), and Forward Access CHannel (CELL_FACH). We omit URA_PCH and CELL_PCH in this study. While URA_PCH plays only a role in scenarios of high mobility, CELL_PCH is not yet widely implemented. Our results are still of general nature and do not depend on the number of considered RRC states.

RRC State Transitions If the UE is switched on and no data connection to the mobile network is established, the UE is in RRC_Idle state. If the UE wants to send data, radio resources are allocated by the NodeB for the handset and the UE will transition to either the CELL_FACH or the CELL_DCH state. Then, a corresponding channel for data transmission is assigned to the UE and the UE is connected to the network. The CELL_FACH and the CELL_DCH state can be distinguished in that way that in CELL_DCH state a high-power dedicated channel for high speed transmission is allocated whereas in CELL_FACH state a shared access channel for general sporadic data transmission is used. Thus, CELL_FACH consumes significantly less power than the CELL_DCH state.

The possible transitions between the different states are defined by the network operator and the RRC protocol stack. Typically, the following state transitions are included: RRC_Idle $\rightarrow$ CELL_FACH, CELL_FACH $\rightarrow$ CELL_DCH to switch from lower radio resource utilisation and low UE power drain to another state using more resources and power, and CELL_DCH $\rightarrow$ CELL_FACH, CELL_FACH $\rightarrow$ RRC_Idle, CELL_DCH $\rightarrow$ RRC_Idle to switch to lower resource usage and power drain. According to [29, 30], the transitions are triggered by user activity and radio link control buffer level at the base station. A transition from CELL_DCH to CELL_FACH usually occurs when



(a) Three State Model



(b) Two State Model

Figure 2.3: RRC state machine diagrams.

the buffer is empty and a threshold for a release timer is exceeded, resulting into the corresponding RRC protocol message flow. A transition in the reverse direction is triggered if the buffer level exceeds a specified threshold value for a predefined time period. The UE will transition into RRC_`Idle` state if the RNC detects overload in the network or no data was sent by the UE for a specified time.

Considered Network Models We consider two different state transition models, depicted in Figure 2.3, based on the RRC protocol. The first model including the RRC_`Idle`, CELL_FACH, and CELL_DCH states is shown in Figure 2.3a and is in the following called the *Three State Model*. If the UE is in the RRC_`Idle` state and activity is detected, i.e. a packet is sent or received, the connection transitions to CELL_DCH state. After each transmission a timer T_{DCH} is started and reset whenever a new packet is sent or received. If the timer expires, the connection transitions to the CELL_FACH state; upon entering, the T_{FACH} timer is started. If a new transmission occurs, the connection again transitions to the CELL_DCH state. If T_{FACH} expires, the connection transitions to RRC_`Idle` state.

The second model, denoted as the *Two State Model*, and shown in Figure 2.3b, only includes the RRC_`Idle` and CELL_DCH state. If the UE is in the RRC_`Idle` mode and a packet is sent or received, the connection transitions to the CELL_DCH state. Once in CELL_DCH mode, the T_{DCH} timer is started and it is reset whenever a new packet is sent or received. If the timer expires, the UE transitions back to RRC_`Idle` state.

Proprietary Fast Dormancy Extensions While the Three State Model is closer to the specified RRC protocol, the Two State Model is similar to some proprietary *Fast Dormancy* implementations used by UE vendors. In these Fast Dormancy implementations, the UE tears down the connection to the network state as soon as no data is ready to be sent for a certain time, i.e., it forces the network to transition to RRC_`Idle` state. In contrast to the Three State Model, there is no transition to the CELL_FACH state. If a device disconnects from the network by transitioning to the RRC_`Idle` state, it has to be re-authenticated before another transition to the CELL_DCH state can occur. This results in additional signalling traffic and causes more load on the network [27] due to frequent re-establishments of the RRC connection. These proprietary Fast Dormancy algorithms do not adhere to the RRC specification [24], but nonetheless exist in the real world and have been identified as possible causes for signalling storms. The major reason for Fast Dormancy implementations is the decrease in power drain on the UE, since the transmission unit of the UE consumes only 1 % to 2 % of the power in RRC_`Idle` state compared to the CELL_DCH state. Thus, both models warrant further investigation.

2.1.2 Measurements of RRC Parameters and Optimisation of Resource Consumption

In the literature the configuration of the inactivity timers used for the RRC protocols have been investigated in detail. In [30] a measurement tool for RRC protocol states is presented. It is used to determine RRC state transition parameters, channel setup delays, and paging delay by measuring the one-way round trip time of data packets. The results are validated by monitoring the power drain in different RRC states. One outcome is the observation that UMTS network configurations vary significantly by network operator. The CELL_DCH release timer as well as the inactivity timer value triggering transition to RRC_`Idle` state were measured. The values range from 1.2 s for the CELL_DCH release timer to more than one minute for the RRC_`Idle` timer. Similar results are presented in [29]. Here, the observed values vary between 5 s and 12 s. Additionally, they also determined the exact RRC state transitions for two networks, i.e. RRC_`Idle` → CELL_FACH → CELL_DCH or RRC_`Idle` → CELL_DCH directly without transitioning through the CELL_FACH state. The 3rd Generation Partnership Project (3GPP) has released a technical report [31] about the adverse impact of mobile data applications. This report states that frequent connection re-establishments due to small data packets caused by e.g. status updates of social network or instant messaging applications can lead to problems of increased signalling. This highlights the importance of this topic.

Furthermore, there are papers that propose optimising strategies that take the RRC states into account. In [26] the impact of different application traffic patterns is studied to reveal resource usage in mobile networks. By identifying packet bursts, they infer the RRC states of the UE. Radio resources are quantified by channel occupation time and power drain. They propose an algorithm that tries to optimise application traffic patterns by e.g. piggybacking, batching up data, or decreasing the update rate of an application. The algorithm is evaluated for six applications: two news applications, the Pandora streaming application, Google search, a Tune-In radio and Mobelix. In [32] RRC states are studied for network optimisation. The authors optimise the inactivity timers to allow a better resource utilisation. They propose an application-to-network interface to avoid unnecessary timer periods after data transmission.

2.1.3 Smartphone Power Consumption and QoE

As discussed at the beginning of this chapter, power drain of the UE and the QoE for the end user are important KPIs for the hardware vendor and the application developer, respectively. Power drain of the UE varies according to the devices' current RRC state. The power drain caused by CELL_DCH mode was measured on specific devices at about 600 mW to 800 mW [26, 29]. In CELL_FACH mode, the power drain of an *HTC TyTN II* was measured at about 400 mW to 460 mW depending on the UE and the network operator [29]. A precise measurement of the power drain of different RRC states is performed in [29, 33, 34]. The authors report that the power drain depends on two factors: *a*) user interactions and applications, *b*) platform hardware and software. Methods for reducing power drain in Long Term Evolution (LTE) Machine to Machine scenarios are considered in [35]. The authors consider tradeoffs between responsiveness and power drain by means of prolonging the discontinuous reception cycles in the LTE standard. The authors of [36] perform a measurement of power drain and RAN signalling during playback of a YouTube video in 3G and LTE UEs. They employ a proxy server in order to ensure that traffic is sent in bursts, thus decreasing power drain at the cost of additional signalling traffic.

In [37] the authors performed a four week long study with 29 participants to identify factors influencing the QoE of mobile applications. The study comprises *a*) data from context sensing software, *b*) user feedback using an experience sampling method several times per day, and *c*) weekly interviews of the participants. To determine the factors of influence, the authors analyse the frequency of specific keywords in the interviews and the surveys. They find that the term *battery* is mentioned by the participants with the highest frequency. According to the authors this is reasonable since the battery efficiency has a strong impact on the user perceived quality, in particular when the UE is nearly discharged.

2.2 Inferring Signalling Frequency and Power Consumption from Network Traces

All participating stakeholders, i.e. network operators, hardware vendors, and application developers, need to assess the impact of potential changes of parts of the mobile network, e.g. change of network parameters, introduction of new hardware or modification of applications, on their considered KPIs without rolling out changes to the production network. To this end, we propose specific metrics in order to quantify the impact of changes on the network on the considered KPI. We introduce an algorithm to infer metrics from application traffic measurements, network parameters and power and signalling configurations. In Section 2.2.1 we present the algorithm and methods to describe the relevant metrics. Then, in Section 2.2.2, we use the proposed methodology to evaluate the impact of various network configurations on four popular applications.

2.2.1 Inferring State Transitions and Deriving Metrics

A UE's firmware triggers RRC state transitions based on application traffic. While solutions exist to capture RRC state transitions on specific hardware [38] they are not available for all modern smartphone platforms. Other options to measure the required information include using costly hardware and use specific UEs, which are usually not available to researchers and application developers. This prevents the developers from evaluating the effect of their applications on the overall health of the network. Consequently, they can not take measures to prevent the harmful behaviour of their applications. However, it is possible to infer the RRC state transitions for a given packet trace if the network configuration is known.

First, we describe the setup used to capture network packet traces for arbitrary apps. Then, we give an algorithm to infer the RRC state transitions for a given packet trace. Based on these state transitions, we can calculate the number of signalling messages generated by the packet trace. Finally, we use the information on when which RRC state was entered to calculate the power drain of the UE's radio interface.

Measurement Procedure and Setup

To investigate the behaviour of the application under study, we capture traffic during a typical use of the application on a *Samsung Galaxy SII* smartphone. The smartphone runs the Android operating system and is connected to the 3G network of a major German network operator. To obtain the network packet traces we use the `tcpdump` application. This application requires *root* privileges which are obtained by rooting the device and installing the custom *cyanogenmod* ROM [1]. Once `tcpdump` is installed and running, we start the application under study and capture packet traces while the application is running. Then, the *android debugging bridge* is used to copy the traces to a workstation. The traces contain Internet Protocol (IP) packets embedded in Linux Cooked Captures. We require the IP packets, which are extracted for use in the following analysis.

Inferring Network State of a UE

In this section we study the influence of the application traffic on RRC state transitions and signalling messages. Since RRC state transitions can not be captured using commonly available tools, we introduce an algorithm to infer RRC state transitions from IP packet traces. Using this algorithm we analyse the RRC state transition frequency and signalling message load for the Two State Model and Three State Model.

Traffic below the network layer can not be measured without specific equipment which interfaces with the proprietary firmware of the UE and is often out of reach for developers interested in assessing the impact of their applications on the network. Based on the Two State and Three State models introduced in Section 2.1.1, we process `tcpdump` captures of the application traffic.

[1] `http://www.cyanogenmod.org`, Accessed: November, 21st 2015

However, it should be noted that this method is not restricted to a specific network model, but can be extended to any other network model as well. Using these captures, we extract the timestamps when IP packets are sent or received. Furthermore, we require the timer values of the transition from CELL_DCH state to CELL_FACH state, T_{DCH}, and the timer for the transition between CELL_FACH and RRC_Idle states, T_{FACH}.

Based on this information Algorithm 1 infers the timestamps of state transitions according to the 3GPP specification [28] for the Three State Model. This algorithm can be simplified to also work for the Two State Model. Alternatively, a method to post-process the results of the algorithm to obtain results for the Two State Model is given at the end of this section. The algorithm first computes the inter-arrival times of all packets. Then, each timestamp is considered. If the UE is currently in RRC_Idle state, a state transition to CELL_DCH occurs at the moment the packet is sent or received. If the inter-arrival time exceeds the T_{DCH} timer the UE transitions to CELL_FACH T_{DCH} seconds after the packet was sent or received. Similarly, if the inter-arrival time exceeds both the T_{DCH} and T_{FACH} timers a state transition to RRC_Idle occurs T_{DCH} seconds after the state transition to CELL_FACH.

Decreasing the power drain of their devices is always a goal of UE vendors. A straightforward way to achieve this, if only the well-being of the UE is considered, is to transition from CELL_DCH state to RRC_Idle as soon as no additional data is ready for sending. While this transition is not directly available in the 3GPP specification for the RRC protocol [28], a UE may reset the connection, effectively transitioning from any state to RRC_Idle. This behaviour can be modelled using the Two State Model introduced in Section 2.1.1.

State transitions for the Two State Model can be calculated using a similar algorithm. Alternatively, the behaviour of the Two State Model can be emulated using Algorithm 1 if T_{FACH} is set to 0 s and all state transitions to CELL_FACH are removed in a post processing step.

Algorithm 1 Inferring RRC state transitions based on IP timestamps.

```
Require: Packet arrival timestamps $ts$
  CELL_DCH to CELL_FACH timer $T_{\text{DCH}}$
  CELL_FACH to RRC_Idle timer $T_{\text{FACH}}$
Ensure: Times of state transition $state\_time$
  New states after state transitions $state$
  interarrival(i) $\leftarrow$ $ts$(i+1) - $ts$(i)
  index $\leftarrow$ 0
  for all ts(i) do
      if state(index) = RRC_Idle then
          index $\leftarrow$ index + 1
          state(index) $\leftarrow$ CELL_DCH
          state_time(index) $\leftarrow$ ts(i)
      end if
      if interarrival(i-1) $> T_{\text{DCH}}$ then
          index $\leftarrow$ index + 1
          state(index) $\leftarrow$ CELL_FACH
          state_time(index) $\leftarrow$ ts(i) $+T_{\text{DCH}}$
      end if
      if interarrival(i-1) $> T_{\text{DCH}} + T_{\text{FACH}}$ then
          index $\leftarrow$ index + 1
          state(index) $\leftarrow$ RRC_Idle
          state_time(index) $\leftarrow$ ts(i) $+T_{\text{DCH}} + T_{\text{FACH}}$
      end if
  end for
```

Table 2.1: Number of signalling messages per RRC state transition perceived at the RNC [28].

From/to	RRC_Idle	CELL_FACH	CELL_DCH
RRC_Idle	–	28	32
CELL_FACH	22	–	6
CELL_DCH	25	5	–

Calculating Signalling Frequency and Power Drain

In reality, the number of state transitions is not the metric of most importance if network signalling is to be evaluated. Each state transition results in a number of RRC messages between the UE and different network components. For this study we consider the number of messages observed at the RNC, which can be found in [28] and is summarised in Table 2.1. It can be seen that transitions from or to the RRC_Idle state are especially expensive in terms of number of messages sent or received. This is due to the fact that upon entering or leaving the RRC_Idle state, authentication has to be performed. Note that for the Two State Model only transitions from or to the RRC_Idle state occur. This results in the fact that for the same network packet trace the number of signalling messages occurring in the Two State Model is generally higher than in the Three State Model. To obtain the total number of signalling messages, we weigh the number of state transitions with the number of messages sent per state transitions. Then, we average the number of state transitions over the measurement duration to obtain a metric for the signalling load at the RNC, i.e. the Signalling frequency (SF). The inference algorithm does not differentiate between state changes caused by upstream or downstream traffic. State changes caused by downstream traffic usually generate some additional signalling messages, as paging is involved. The inference algorithm can easily be enhanced to support this behaviour. However, the results discussed in the next section would only change quantitatively. Furthermore, the algorithm can be adapted to new networking models or other numbers of signalling messages sent per

Table 2.2: Power consumption of the UE radio interface depending on current RRC state [26].

RRC State	Power consumption
RRC_Idle	0 mW
CELL_FACH	650 mW
CELL_DCH	800 mW

state transition.

From a user's point of view, the signalling message frequency is of little importance. The user is interested in a low power drain as this increases the battery life of the device. To calculate the battery life, we use the time when state transitions occurred, and the new state, to calculate the relative amount of time that was spent in each state. Given the relative time spent in each state, we use Table 2.2, taken from [26], to compute the Power drain (PD) of the radio interface during the measurement phase. We focus on the power drain of the radio interface, as it is possible to measure the aggregated power drain using out-of-the-box instrumentation techniques provided by the hardware vendor.

2.2.2 Impact of Application Traffic Patterns

In the measurement study, we apply the methods introduced in Section 2.2.1 to four popular smartphone applications to infer signalling traffic and power drain. First, we characterise the applications in terms of traffic patterns, application usage, as well as bandwidth requirements. Then, we study the SF and power drain caused by these applications if the inactivity timers, i.e. T_{DCH} or T_{FACH} are modified. Finally, we analyse the influence of network parameters on web QoE in terms of Mean Opinion Score (MOS) depending on page load times which are influenced by the network settings.

Table 2.3: *Qualitative characterization of applications under study.*

Application	Traffic characteristic	Application use	Required bandwidth
Angry Birds	Interactive	Foreground	Low bandwidth
Aupeo	Interactive	Background	High bandwidth
Twitter	Periodic, Low frequency	Background	Low bandwidth
Skype	Periodic, High frequency	Background	Low bandwidth

Characterisation of Traffic Patterns for Selected Applications

For this study we chose four specific applications in order to cover a broad spectrum of traffic characteristics, as described in Table 2.3. First, we discuss said characteristics for these applications. We differentiate between applications, where the user interaction causes the generation of traffic, and those, where the application periodically sends or receives traffic. Finally, we consider the amount of bandwidth used by the application.

Angry Birds for Android is a popular *interactive* free-to-play game and runs in the *foreground*. To finance the game, an advertisement is shown once the player starts or restarts a level. Advertisements are downloaded on demand by the application, but require *low bandwidth*. Thus, the time between two advertisements depends on the frequency of the player advancing to the next level or deciding to restart the current one.

Aupeo is an Internet radio application, allowing a user to listen to content from personalised radio stations, while running in the *background*. Content is not streamed but downloaded at the beginning of the track. The exact duration depends on the radio stations chosen by the user and is thus *interactive*. This results in large times of inactivity during the playback of the track itself. Due to the fact that audio files are downloaded, there is a *high bandwidth* requirement.

The **Twitter** client is used to send and receive new short messages from the user's Twitter account. Transferring these messages requires relatively *low bandwidth*. To this end, the user can specify an update frequency when to pull new messages in the *background*. Thus, the downloads occur with a *periodic behaviour of low frequency*, where the client sends an HTTP Over TLS (HTTPS) request to the Twitter server and in return receives new Tweets for the user's account. We do not consider an active user who is publishing new Tweets. Such behaviour would manifest as additional traffic to the periodic one generated by the status updates. Due to the fact that publishing updates occurs relatively infrequently, and updating the feed occurs more often, the traffic generated by publishing updates is dominated by that occurring due to updates, and thus can be neglected.

Finally, we consider the **Skype** application. We do not consider any Voice over IP (VoIP) calls, but the application's idle behaviour, i.e. when the application is running in the *background*. During this time, the application sends keep-alive messages to the network. These keep-alive messages are sent with a *high frequency* and require *low bandwidth*.

In addition to the applications considered, there exist other categories of applications which are running in the *foreground* and *interactively* require a *high bandwidth*. One example for such an application is Skype while taking a VoIP call. These applications are not considered in this study, as this kind of behaviour causes the UE to be always online. This results in the minimal amount of signalling messages to be sent and a maximal power drain at the UE, independent of network model or used parameters. Other combinations of traffic criteria also exist. However, from both, a signalling frequency as well as a power drain point of view, they can be mapped to one of the discussed cases. For example, if an application is sending periodic updates with low bandwidth without user interaction, then the fact that the application is running in the foreground or the background is without consequence for the generated signalling frequency or power drain. However, these cases should be considered when optimisation strategies for message sending are under study. Background applications, for instance, could allow for the batching of messages, due to the fact that the transmission is usually not urgent, while foreground applications do not allow for such behaviour as it would delay the user interactions and consequently de-

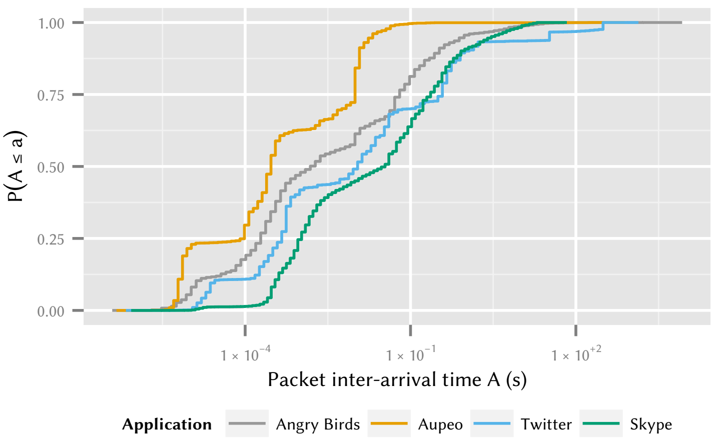


Figure 2.4: CDF of inter-arrival times for considered applications.

crease QoE.

Next, we describe the applications under study in more detail. For each application we show the Cumulative Density Function (CDF) of the inter-arrival times in Figure 2.4 and give information about the mean values and standard deviation of both inter-arrival times and bandwidth in Table 2.4, respectively.

a) **Angry Birds** We see that there are no distinct peaks in inter-arrival time, which would indicate a periodic behaviour. Furthermore, we see that $5\,\%$ of all inter-arrival times are greater than $1\,\mathrm{s}$. As we consider only T_{DCH} values above $1\,\mathrm{s}$, those are candidates for triggering state transitions. The mean inter-arrival time is $0.66\,\mathrm{s}$, with a relatively high standard deviation of $15.90\,\mathrm{s}$. This is caused by the low inter-arrival times in one advertisement request at the beginning of each new level and the relatively large inter-arrival times between two advertisements. Mean bandwidth is relatively low with $4.42\,\mathrm{kbit\,s^{-1}}$ and a high standard deviation

Table 2.4: Mean and standard deviation of inter-arrival time and bandwidth for considered applications.

Application	Inter-arrival time (s)		Bandwidth (kbit s^{-1})	
	Mean	Standard deviation	Mean	Standard deviation
Angry Birds	0.66	15.90	4.42	4.50
Aupeo	0.06	3.06	129.76	482.63
Twitter	8.91	44.09	0.27	0.04
Skype	0.55	1.95	1.30	1.84

of $4.5\,\mathrm{kbit\,s^{-1}}$. These differences can be explained by considering the behaviour of the application. During long phases of use no traffic is sent, and after a level is restarted, a new advertisement has to be obtained, causing the transmission of data. Note that no level data is downloaded during gameplay at all, as the complete game is downloaded during the installation process.

b) **Aupeo** We see that the application generates packets with relatively small inter-arrival times with a small mean inter-arrival time of $0.06\,\mathrm{s}$. The high standard deviation of $3.06\,\mathrm{s}$ is caused by the waiting between two tracks. Furthermore, we see a high mean bandwidth of $129.76\,\mathrm{kbit\,s^{-1}}$, and a standard deviation of $482.63\,\mathrm{kbit\,s^{-1}}$. This is caused by the difference in traffic activity between times when tracks are either downloaded or not.

c) **Twitter** We see that $90\,\%$ of all transmissions occur with an inter-arrival time of less than $1\,\mathrm{s}$. Also, we can observe a high mean inter-arrival time of $8.91\,\mathrm{s}$ and a high standard deviation of $44.49\,\mathrm{s}$. Additionally, the mean bandwidth is low with only $0.27\,\mathrm{kbit\,s^{-1}}$ and a low standard deviation of $0.04\,\mathrm{kbit\,s^{-1}}$ due to the fact that Twitter text messages are only 140 characters in length and thus only a low volume of traffic needs to be transmitted.

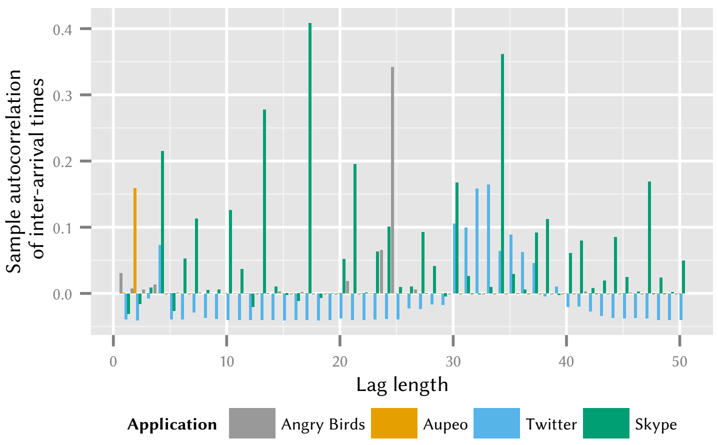


Figure 2.5: Autocorrelation of inter-arrival times for considered applications.

d) **Skype** Similar to the Twitter application, we see that 90 % of all packets occur with an inter-arrival time of less than 1 s. However, in contrast to Twitter, we see a low mean inter-arrival time of 0.55 s with a standard deviation of 1.95 s. Further, we observe a relatively low mean bandwidth of 1.30 kbit s^{-1} and a standard deviation of 1.8 kbit s^{-1}.

To further study the traffic patterns of the applications, we study the autocorrelation of the packet inter-arrival time with regard to the lag length in Figure 2.5. We note that all studied applications present completely different autocorrelations for the inter-arrival times. This is one of the reasons that the applications under consideration will display different signalling behaviour in the next section.

Influence of Application Characteristics on Optimisation with Network Timers

This section studies the impact of traffic generated by applications on both the network and the QoE of the user. We consider two metrics. First, we consider the frequency of signalling messages induced at network components in the RAN. In light of network outages caused by so called signalling storms, a large number of signalling messages leading to overload at network equipment, it is in the interest of a network operator to reduce the number of signalling messages arriving at the RNC. One possible way to reduce the signalling frequency SF is to modify network timer values, i.e., T_{DCH} and T_{FACH}.

As discussed in Section 2.1.3, the QoE a user perceives while using the device is influenced by the battery life of the UE. Thus, the second metric considered is the device's power drain which is influenced by the used network model and associated timer settings. As described in Section 2.2.1, based on a measurement trace for an application we use Algorithm 1 to infer the state transitions occurring during the use of the application. Then, we calculate the relative time spent in each state and use Table 2.2 to compute the mean power drain of the radio interface during the measurement. We study both metrics, first on their own and then aggregated for both network models introduced in Section 2.1.1.

In this section we first consider the Three State Model, which describes the default behaviour in 3G networks. Then, we describe the influence of the Two State Model which models a network behaviour similar to that if proprietary fast dormancy algorithms are used. These algorithms have been identified as one of the causes of a signalling storm [27]. Finally, we summarise the results and discuss the possible ramifications of using network timer values to reduce the signalling frequency.

Three State Model: Signalling Frequency vs. Power Consumption First, we investigate the signalling frequencies generated by the studied applications for the Three State Model. Figure 2.6 shows the signalling frequency SF with regard to the T_{DCH} timer. For all studies of the Three State Model, the T_{FACH}

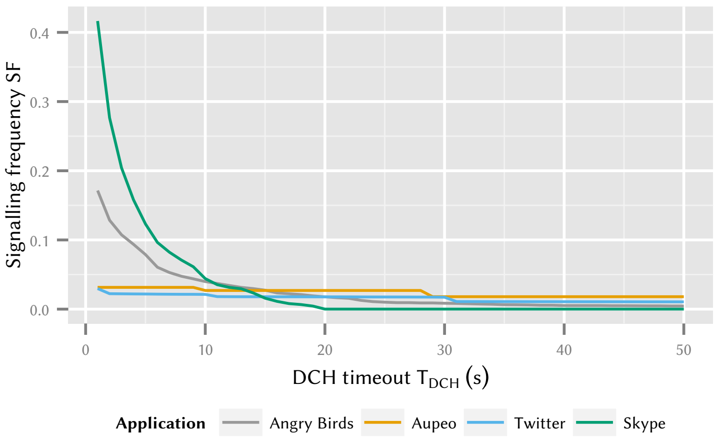


Figure 2.6: Signalling frequency SF for varying T_{DCH} timers for the Three State Model.

timeout is set to $T_{\mathrm{FACH}} = 2 \cdot T_{\mathrm{DCH}}$, a realistic value as shown in [26]. We see that for T_{DCH} timers shorter than 6 s the Skype application in RRC_`Idle` mode generates the highest signalling frequency. The Angry Birds application generates the second highest frequency of signalling messages, followed by the Aupeo application. The Twitter application generates the smallest signalling frequency. If the T_{DCH} value is longer than 15 s, this order changes. However, in general the signalling frequency for higher T_{DCH} timeouts is lower than for shorter T_{DCH} timeouts. Now, the Aupeo application has the highest signalling frequency, followed by the Twitter application. The signalling frequency for the Angry Birds application takes the third place. The application which generated the highest signalling frequency generates the lowest frequency for higher timeout values. This behaviour can be explained by the fact that the Skype application sends keep-alive messages with an interval of less than 20 s. If the timer is greater than the interval time of the keep-alive messages, the UE stays always connected and thus generates almost no signalling.

These results show that the traffic patterns of the application have a large influence on the generated signalling frequency. Signalling is generated for every pause in sending or receiving larger than the configured timeouts. If such pauses occur frequently, this increases the signalling frequency as shown on the examples of Skype and Angry Birds. Applications with more time between sending or receiving of data cause less signalling, as shown by Aupeo and Twitter. Furthermore, we can observe that the signalling frequency can be reduced by increasing the T_{DCH} timeout, with the minimum being reached as T_{DCH} approaches infinity. From a signalling frequency perspective, a value of 20 s would probably be sufficient, however if other metrics, e.g. radio resource consumption, are considered 10 s would be acceptable for a network operator.

Based on this finding, we see that increasing the T_{DCH} timer decreases the signalling frequency SF at the RNC. However, the actual signalling frequency depends on the application running at the UE. From a network operator's point of view, the Three State Model should always be preferred to the Two State Model as it generates less signalling messages per second, thus decreasing the load at the RNC. This view does however not consider the additional radio resources which are kept in use for a longer time if larger T_{DCH} values are used. Additionally, it should be noted that the choice of the network model is sometimes outside of the domain of the network operator. Proprietary Fast Dormancy algorithms, as the considered Two State Model, are enabled on the UE by the user.

In Figure 2.7 we consider the power drain if the network uses the Three State Model, i.e. if the Fast Dormancy mode of the UE is disabled. The figure shows the mean power drain PD of the device with regard to the T_{DCH} timeout. Possible values range between 0 mW, if the UE was in RRC_`Idle` state during the whole measurement, and 800 mW, if the UE was in CELL_DCH state during the complete measurement. We see that the lowest power over all considered T_{DCH} values is consumed by the Twitter application. The second least power drain is required by Aupeo, followed by Angry Birds. Finally, the most power is consumed by Skype. Here we see that the maximum value of 800 mW is reached

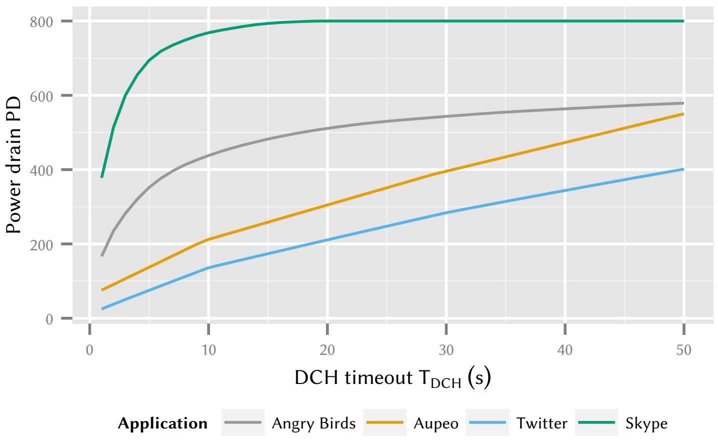


Figure 2.7: Power drain PD for varying T_{DCH} timers for the Three State Model.

at a T_{DCH} timeout of 20 s. Due to the periodic traffic behaviour of Skype, the device is always in CELL_DCH state. Again we see that the traffic characteristics of the applications impact the power drain. Applications with more network activity are forced to stay in connection states requring a larger amount of power for a longer time. We see that for very small network timers, the power drain is minimal. However, as seen in the last section small timers increase the signalling frequency at the RNC. Again, a choice of 10 s for the T_{DCH} timer can be seen as a compromise between signalling frequency SF and power drain PD.

Finally, we aggregate both metrics in in Figure 2.8. The x-axis of the figure gives the signalling frequency. On the y-axis we show the power drain PD. Different T_{DCH} values are shown by different colours as specified by the colour bar. First, we consider Angry Birds. We observe that as the signalling frequency approaches zero, the power drain rapidly increases, even if only small gains in signalling frequency reduction can be achieved. The Aupeo application presents a completely different picture. Here, we can see multiple almost horizontal lines

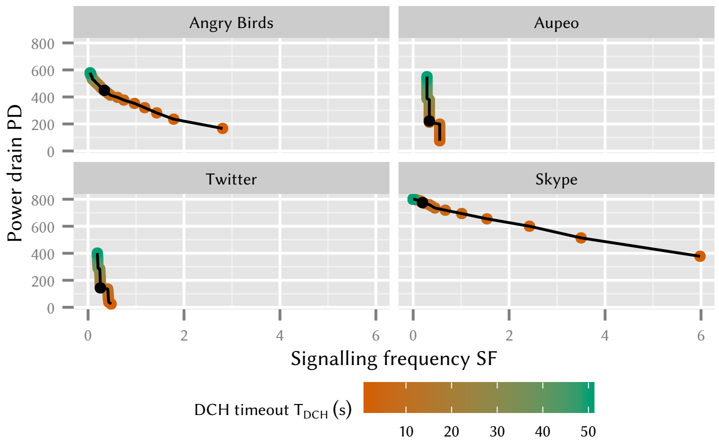


Figure 2.8: Influence of manipulating T_{DCH} timer on signalling frequency SF and power drain PD for the Three State Model. Filled marker highlights $T_{DCH} = 11\,\mathrm{s}$.

of markers. If T_{DCH} is chosen in this range, each increase of T_{DCH} brings a small decrease in signalling frequency SF for an increase in power drain PD. However, some points of discontinuity exist. If for example the CELL_DCH timer is increased from 10 s to 11 s, a decrease in signalling frequency SF of 40 % can be achieved by only suffering from a small increase in power drain. These points of discontinuity would present themselves to be suitable targets of optimisation. Next, we consider the Twitter application. It displays a similar behaviour as the Aupeo application, with multiple points of discontinuity. Note that Twitter exhibits a different point of discontinuity, and the T_{DCH} value of 10 s, which provided good results for Aupeo, is not optimal for Twitter. Finally, Skype again shows a completely different picture than all the other considered applications. First, note that due to the large signalling frequency SF of Skype for small values of T_{DCH}, $T_{\text{DCH}} = 1\,\text{s}$ is not displayed in the figure. Furthermore, as the T_{DCH} timer increases above 20 s the signalling frequency SF does not decrease any further, and the power drain PD remains at the maximum value. We observe that there is no common optimal value for all applications which would result in an acceptable tradeoff.

Two State Model: Signalling Frequency vs. Power Drain Now, we study the consequences of the application traffic in a network using the Two State Model. The Two State Model occurs in reality if Fast Dormancy implementations are considered. Here, the UE disconnects from the network if for a certain time no traffic is sent or received in order to reduce power drain. As for the Three State Model, Figure 2.9 shows the signalling frequency SF with regard to the setting of the T_{DCH} timer. We see the same general behaviour as with the Three State Model, however the signalling frequencies generated by each of the applications for the Two State Model are usually higher. For example, even for relatively high T_{DCH} timeout values of 10 s, the Angry Birds application causes 270 % of the signalling frequency SF with respect to a network using the Three State Model.

Next, we consider the changes in the power drain of the UE if the user decides

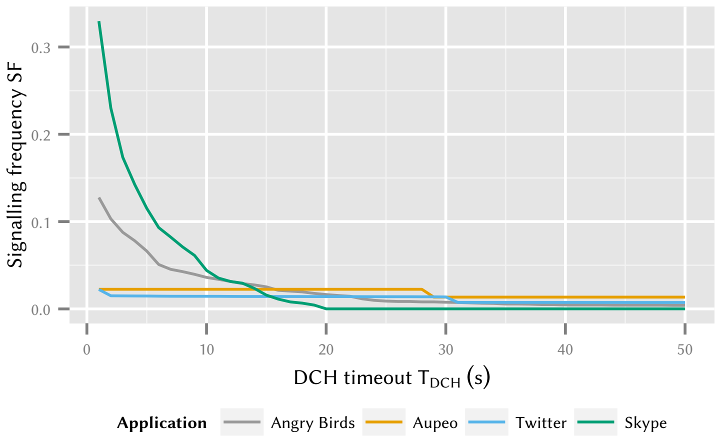


Figure 2.9: Signalling frequency SF for varying T_{DCH} timers for the Two State Model.

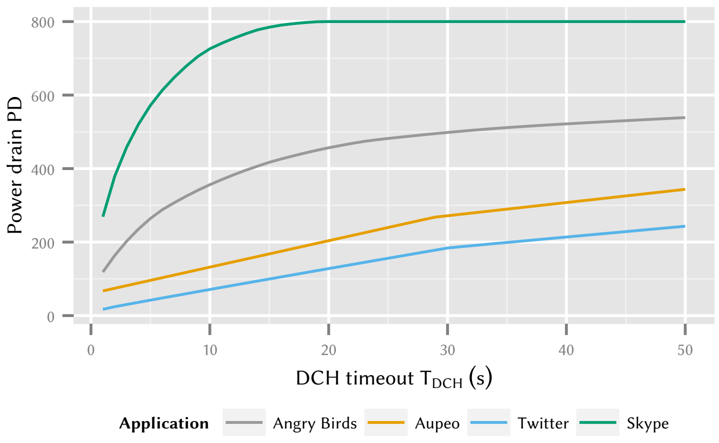


Figure 2.10: Power drain PD for varying T_{DCH} timers for the Two State Model.

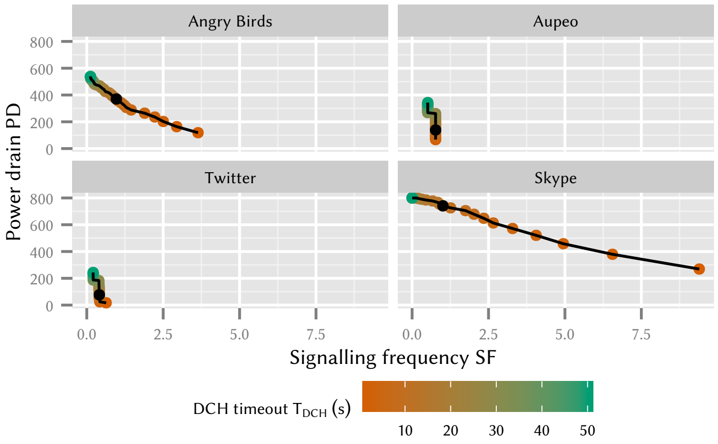
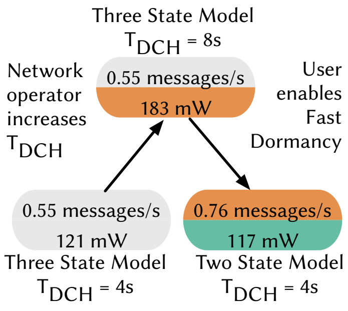


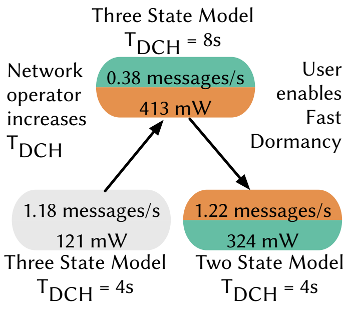
Figure 2.11: Influence of manipulating T_{DCH} timer on signalling frequency SF and Power drain PD for the Two State Model. Filled marker highlights $T_{DCH} = 11\,\mathrm{s}$.

to enable Fast Dormancy, i.e. switch to a Two State Model, in Figure 2.10. As with the signalling frequency, we only see a quantitative difference to the Three State Model. Again, we compare the differences between Two State Model and Three State Model on the example of the Angry Birds application. For the same considered T_{DCH} timeout of 10 s, we see a decrease of 81 % in power drain PD when compared with the Three State Model.

Finally, we compare the influence of changes of the T_{DCH} timeout on both signalling frequency SF and power drain PD for the Two State Model in Figure 2.11. As for the Three State Model, we see that there is no tradeoff between power drain and signalling frequency which would be acceptable for all application. Even for single applications T_{DCH} values, as for example the earlier discussed 10 s, which for Angry Birds was an acceptable tradeoff, is no longer a good choice in the Two State Model.



(a) Aupeo



(b) Angry Birds

Figure 2.12: Influence of manipulating T_{DCH} timer on different applications.

Consequences of Trade-Off: Signalling Frequency vs. Power Drain

In order to illustrate the impact of the behaviour discussed in the previous section, we compare the influence of the T_{DCH} timer on two applications with different traffic characteristics in Figure 2.12.

First, we consider the Aupeo application as shown in Figure 2.12a. The signalling frequency SF before the increase of the T_{DCH} timer was 0.55 messages per second, after the change to $T_{\mathrm{DCH}} = 8\,\mathrm{s}$ the signalling frequency remains unchanged. Thus, the policy change based on one application brings no significant gain to other applications. However, from a user's point of view, the power drain PD increased from $121\,\mathrm{mW}$ to $183\,\mathrm{mW}$. Again, we assume the user activates fast dormancy to deal with the increase in power drain of more than 50%. This results in a decrease of power drain PD to $117\,\mathrm{mW}$, and an increase of overall signalling frequency SF to 0.76 messages per second. By changing the value without considering all applications, the network operator reduces the QoE for other users, and worsens the overall situation. Thus, due to the large number of applications it seems impossible to optimise the T_{DCH} timeout to reduce the signalling frequency without negatively impacting the users QoE in unexpected ways. The Angry Birds application shown in Figure 2.12b shows quantitatively similar results.

There exist applications, like Twitter and Aupeo, where optimisation by modifying the T_{DCH} values can provide acceptable results. However, these optimisations are only successful if a single application or network model is considered. For other applications, like Angry Birds or Skype, this optimisation approach does not seem to be successful. A reduction of signalling frequency and power drain is possible, if the application developers are incentivised to optimise their applications accordingly. In [26] the authors suggest methods to achieve this optimisation, for example batch transfer of advertisements for applications like Angry Birds or decreasing the refresh rate in applications like Skype. However, at the moment application developers are neither receiving incentives to optimise applications in this way, nor do hardware vendors provide interfaces to facilitate such optimisation. Such interfaces would allow application developers to schedule their data transmissions in such a way that both signalling and battery drain would be reduced. Additionally, these interfaces would need to allow the application developer to specify whether sending the transmission is urgent. One example of such urgency would be if the application is being actively used by the user and requires the feedback of the transmission. If the data is being sent as a regular update while the application is running in the background it could be scheduled for later transmission as suggested by [39, 40].

2.2.3 Influence of Network Configuration and Background Traffic on Web QoE

So far we have discussed only power drain as a QoE influence factor. For applications like web browsing, one relevant QoE influence factor are page load times. Therefore, we consider a web QoE model which quantifies the impact of page load times on mean opinion scores [41]. Here we distinguish between *web QoE* and *QoE*, as no QoE models are currently existing which consider page load times as well as power drain. In this section, we study the impact of background traffic as well as network timer settings on the page load time of an image and the resulting MOS. For this study, we only consider the Three State Model, but the results can be applied to the Two State Model as well.

We assume a scenario, where a user is running a background application like Twitter or Skype. Then, while the application is in the background, the user begins to download an image from a website. Due to the background traffic, and depending on the network model and associated timer values, the UE may be currently either in RRC_`Idle`, CELL_FACH or CELL_DCH state. We give the probability of a random observer encountering the system in CELL_FACH state by $p_{\text{CELL\_FACH}}$ and the probability of a random observer encountering in RRC_`Idle` state by $p_{\text{RRC\_Idle}}$. If the device is currently not in CELL_DCH state, it takes some time to connect. This promotion time depends on the current state and is according to [32] $2.5\,\text{s}$ if the UE is in RRC_`Idle` state and $1.5\,\text{s}$ if the device is in CELL_FACH state. For this study, we assume that the user randomly chooses a time to begin downloading an image. The time until the image is displayed consists of the time to load the page t_p, as well as the time to go online t_o, where t_o is the mean time to go online, given as

$$t_o = p_{\text{RRC\_Idle}} \cdot 2.5\,\text{s} + p_{\text{CELL\_FACH}} \cdot 1.5\,\text{s}.$$

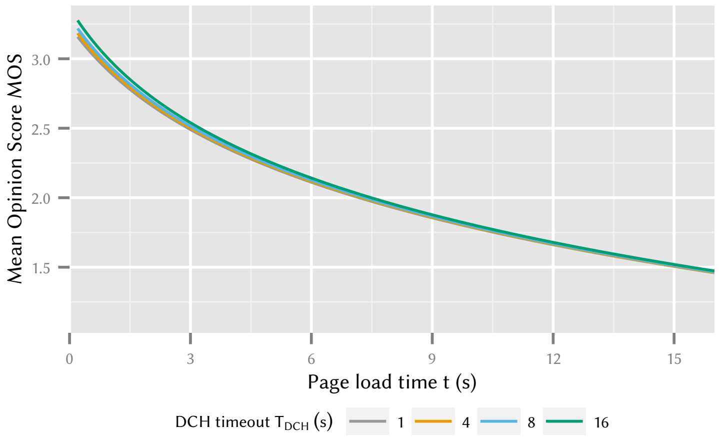
In reality, an additional delay is added due to the latency of the physical display, however as this happens in a smaller timescale we neglect it in this model. Thus, the total time t that is required to download the image is given by $t = t_o + t_p$.

The authors of [41] give a function to calculate the MOS based on the required page load time as $QoE(t) = a \cdot \ln t + b$, were a and b depend on the type of content being downloaded. For our scenario, picture download, values of $a = -0.8$ and $b = 3.77$ are suggested. It has to be noted that for different web sites, the logarithmic function was still observed, but different values for a and b were obtained as given in [41]. These values depend for example on the type of web page as well as the size of the content. Nevertheless, the results presented in this section are therefore generalisable for web browsing to various pages. This allows us to give an expected MOS for downloading pictures while a background application is influencing the probability of a device already being in CELL_DCH state or still having to be promoted to CELL_DCH state.

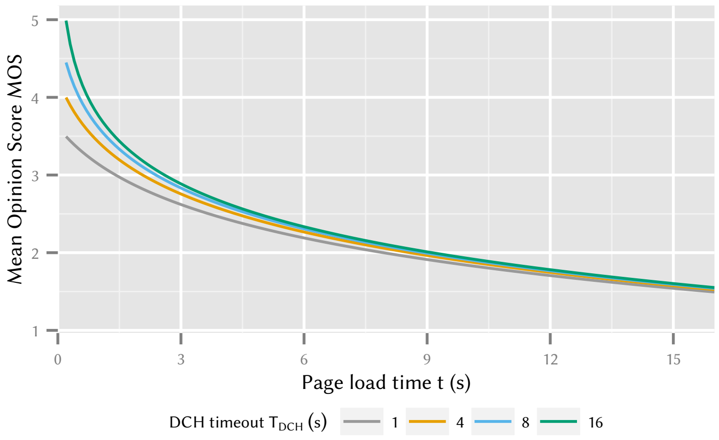
Using this methodology, we study the influence of background traffic on the QoE for two background applications with different traffic characteristics. In Figure 2.13a we assume that the user is running the Twitter application as a background process. The application is set to update the user's status feed every 300 s. In Figure 2.13b the user is running the Skype application as a background application. This application sends keep alive messages every 20 s. For each application, we assume the Three State Model with T_{DCH} settings of 1 s, 4 s, 8 s and 16 s. We always set $T_{\text{FACH}} = 2 \cdot T_{\text{DCH}}$. In both figures we show the assumed page load time t, as provided by the network, on the x-axis for values from 0.2 s to 25 s. We assume 0.1 s as a lower bound, as page load times lower than 0.1 s seconds are not distinguishable [42] by humans. The calculated MOS values are given on the y-axis.

The picture downloads with the background traffic generated by the Twitter application result in MOS values beginning at 3.15 for $T_{\text{DCH}} = 1$ s, 3.18 for $T_{\text{DCH}} = 4$ s, 3.21 for $T_{\text{DCH}} = 8$ s, and 3.27 for $T_{\text{DCH}} = 16$ s respectively. With increasing page load time, the MOS decreases again. This behaviour is due to the fact that the Twitter application periodically sends traffic every 300 s. Then, no further activity occurs until the next refresh occurs. In this time, the UE transitions to RRC_Idle state. This traffic characteristic causes a high probability of a user encountering the device in an RRC_Idle state. Additionally, the traffic characteristics of the background application show that different T_{DCH} settings impact the web QoE only marginally, resulting in the lines in the graph being grouped close together.

In contrast, downloading pictures with the Skype application generating background traffic, causes different MOS values. For a page load time t of 0.2 s the MOS value with $T_{\text{DCH}} = 1$ s is 3.49, with $T_{\text{DCH}} = 4$ s we get 3.99, for $T_{\text{DCH}} = 8$ s we get a MOS value of 4.44, and finally for $T_{\text{DCH}} = 16$ s we get 4.99 respectively. We observe, that for increased page load times, the MOS de-



(a) Background traffic generated by Twitter.



(b) Background traffic generated by Skype.

Figure 2.13: Perceived web QoE for loading a page with existing background traffic.

creases. Further, due to the high frequency of traffic sent by the Skype application increased MOS values occur when compared to those of the Twitter scenario. Here, every 20 s traffic is sent. This means that even for relatively low values of T_{DCH} the user has a high probability of encountering a state where no promotion delay is required before the actual page load time t can begin.

From these studies we can conclude that, when considering the QoE on mobile devices, not only the page load time t caused by the network but also additional delays caused by the state of the device should be considered. As shown on two examples, this state can be affected by other applications which are running in the background and generate traffic.

2.3 A Performance Model for 3G RRC States

The algorithm introduced in Section 2.2 can be used to infer the signalling frequency, power drain, and QoE of existing or prototyped applications. However, in order to study the general impact of traffic in a comprehensive way, methods to derive said metrics from analytical traffic distributions are of interest.

In Section 2.3.1 we introduce a model allowing us to analyse both theoretical and empirical traffic models. Then, in Section 2.3.2 we use this model to study the impact of traffic characteristic on metrics, i.e. signalling intensity and power drain.

2.3.1 Analytical Model

This section introduces a performance model for quantifying power drain against signalling load. The model allows researchers and application developers to evaluate analytical and empirical traffic distributions, deriving metrics for signalling and power drain in order to predict the impact of yet unimplemented applications or planned network configurations.

After presenting the system description, we derive the state distribution and the average frequency of state transitions for a Two State Model, e.g. for proprietary fast dormancy implementations of smart-phone vendors [27]. Afterwards, we extend the model to include CELL_FACH for regular 3G networks. Finally, we define comprehensive metrics for signalling load and power drain.

Mobile RAN System Description

We consider a UE which sends and receives a sequence of data packets via a 3G UMTS network. As discussed in Section 2.1.1, the arrival process of the packet transmissions determines the RRC states of the UE. However, the direction of packets, i.e. whether they originate from the UE or the NodeB, has no impact on the RRC states, as the states solely depend on traffic activity. Due to the high impact of RRC states on traffic patterns, we do not consider packet sizes in this model. In real UMTS networks very small packets might be treated differently for RRC states, but we neglect this both for simplicity reasons and as the impact of packet sizes is highly network operator specific [29]. Furthermore, RRC state transitions are complex procedures depending on implementation details of the UE, the specific UMTS release, and the configurations by the network operator. In order to keep our model simple, but realistic, we reduce the set of standardised RRC states and the state transition triggers in the following ways.

In a first step we consider the Two State Model: RRC_Idle and CELL_DCH as shown in Figure 2.3b. The UE switches to CELL_DCH to transmit or receive data and after an inactivity period of duration T_{DCH}, it switches back to RRC_Idle. The motivation for the two states RRC scenario is twofold. First, it serves illustration purposes. We derive the model step-by-step in this simple scenario to explain the ideas behind the equations. Then, the ideas can be easily transferred to the more complex Three State Model. Second, the scenario is of practical relevance since proprietary implementations of the fast dormancy concept can be modelled as the Two State Model, as discussed in Section 2.1.1. Furthermore, this model is very similar to the one found in LTE systems. In LTE, only a distinction between connected and disconnected states can be found, which maps to the RRC_Idle and CELL_DCH states discussed in this model.

In our model we aggregate both packets sent and received by the UE in the

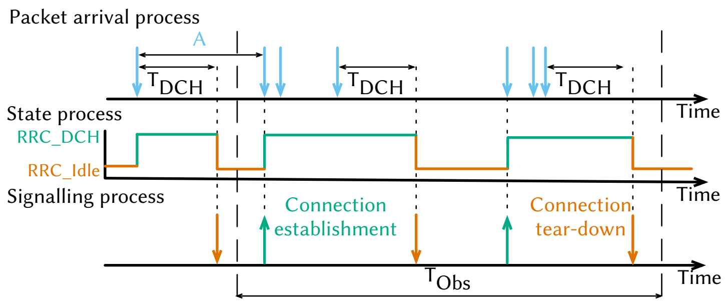


Figure 2.14: Relation of packet arival process A, state process, and signalling process in the Two State Model.

packet arrival process, which is assumed to be a renewal process, i.e. a process with independent and identically distributed (iid) inter-arrival times, described by the random variable A as shown in Figure 2.14. Thus, the probability that the time between two consecutive packets is at most t is $P(A \leq t) = A(t)$. This assumption is validated in Section 2.3.2 using the application measurements obtained in Section 2.2.2.

The packet arrivals determine the RRC state of the UE and the corresponding transitions. Therefore, the packet arrival process can be seen as a modulating process, c.f. [43, 44], while the state and the signalling process represent modulated, i.e., resulting processes.

Two State Model

First, we derive the connection state distribution of the Two State Model and obtain the average frequency of state transitions given a specified packet arrival process.

Connection State Distribution First, we are interested in the state distribution $P(S = s)$, i.e., the fraction of time the UE spends in state $s \in$

$\{\text{RRC\_}\texttt{Idle}, \text{CELL\_DCH}\}$ for a given inter-packet time A. For this purpose, we define an observation interval T_{Obs}, depicted in Figure 2.14, which is assumed to be orders of magnitude larger than the average packet inter-arrival time $E[A]$. In addition, we take the position of an outside observer who observes the state S at a random point in time t^*, uniformly distributed within the observation interval. Then the state distribution $P(S = s)$ is the probability that the observer encounters the UE in state S at the time t^*.

We calculate this distribution as

$$P(S = s) = \int_0^\infty q(\tau) \cdot P(S = s|A = \tau) d\tau, \tag{2.1}$$

where $q(\tau)$ is the probability density that t^* falls into an interval of length τ and $P(S = s)|A = \tau)$ is the probability that the UE is in state S under the condition that t^* is within an interval of length τ.

First, we derive $q(\tau)$. This probability density has to be proportional to $a(\tau)$ and to τ, where $a(\tau)$ is the probability density function of the random variable A. Therefore, we have that $q(\tau) = a(\tau) \cdot \tau \cdot c_0$ with the proportionality constant c_0. Due to $\int_0^\infty q(\tau) d\tau = 1$, we have $c_0 = 1/E[A]$, which leads to

$$q(\tau) = \frac{a(\tau) \cdot \tau}{E[A]}.$$

Next, we derive the conditional probability $P(S = s|A = \tau)$ that t^* falls within a period with state S under the condition that the inter-packet time is $A = \tau$. We use the fact that t^* is uniformly distributed within τ and calculate the probability $P(S = \text{RRC\_}\texttt{Idle})$ by considering the relevant cases:

$$P(S = \text{RRC\_}\texttt{Idle}|A = \tau) = \begin{cases} 0, & \text{if } T_{\text{Obs}} \leq T_{\text{DCH}} \\ \frac{\tau - T_{\text{DCH}}}{\tau}, & \text{otherwise.} \end{cases} \tag{2.2}$$

Similarly, we obtain $P(S = \text{CELL\_DCH})$ as:

$$P(S = \text{CELL\_DCH}|A = \tau) = \begin{cases} 1, & \text{if } \tau \leq T_{\text{DCH}} \\ \frac{T_{\text{DCH}}}{\tau}, & \text{otherwise.} \end{cases} \tag{2.3}$$

Average Frequency of State Transitions Next, we estimate the average frequency of state transitions resulting from a given packet arrival process. For that purpose, we consider again the observation interval T_{Obs} and focus on the state transitions from RRC_`Idle` to CELL_DCH since every switch from CELL_DCH to RRC_`Idle` results in a switch vice-versa. The expected number of observed packets during T_{Obs} is $E[n_{\text{P}}] = T_{\text{Obs}}/E[A]$. Furthermore, the probability that the time between two consecutive packets exceeds the timer T_{DCH} is

$$P(A > T_{\text{DCH}}) = 1 - P(A \leq T_{\text{DCH}}) = 1 - A(T_{\text{DCH}}). \tag{2.4}$$

The number of state transitions $n_{\text{RRC\_Idle}\to\text{CELL\_DCH}}$ during T_{Obs} directly corresponds to the number of inter-packet times exceeding T_{DCH} since an active connection is torn down after an inactivity period of T_{DCH}. Thus, the expected number is

$$\begin{aligned} E[n_{\text{RRC\_Idle}\to\text{CELL\_DCH}}] &= E[n_{\text{P}}] \cdot P(A > T_{\text{DCH}}) \\ &= \frac{T_{\text{Obs}}}{E[A]} \cdot (1 - A(T_{\text{DCH}})). \end{aligned}$$

Hence, the expected frequency of state transitions is

$$E[n_{\text{RRC\_Idle}\to\text{CELL\_DCH}}] = \frac{1 - A(T_{\text{DCH}})}{E[A]}.$$

The same holds also for the state transitions from CELL_DCH to RRC_`Idle` and hence $E[f_{\text{CELL\_DCH}\to\text{RRC\_Idle}}] = E[f_{\text{CELL\_DCH}\to\text{RRC\_Idle}}]$ holds.

Three State Model

In this section we consider three states: RRC_`Idle`, CELL_DCH, and CELL_FACH. Again, we assume that the UE switches from RRC_`Idle` to CELL_DCH whenever it transmits or receives data. After an inactivity of T_{DCH} the UE switches to CELL_FACH, and after an additional inactivity of T_{FACH}, it switches to RRC_`Idle`, as depicted in Figure 2.3a. This scenario usually occurs when the network controls the RRC state of the UE without proprietary connection tear-down mechanisms implemented on the UE. In today's network some operator transition the UE to a state with a paging channel URA_PCH instead of the RRC_`Idle`, but the resource consumptions in both states are very similar and we therefore omit the URA_PCH state for the sake of simplicity.

Connection State Distribution The state distribution $P(S = s)$ for the three states $s \in \{\text{RRC\_}\texttt{Idle}, \text{CELL\_FACH}, \text{CELL\_DCH}\}$ can be derived in the same way as for the scenario with two states. Therefore, we present only the conditional probabilities, which differ from the Two State Model, and use Equation 2.1 for the calculation of the distribution. First, we consider $S = \text{RRC\_}\texttt{Idle}$:

$$P(S = \text{RRC\_}\texttt{Idle}|A = \tau) = \begin{cases} 0, & \text{if } \tau \leq T_{\text{DCH}} + T_{\text{FACH}} \\ \frac{\tau - (T_{\text{DCH}} + T_{\text{FACH}})}{\tau}, & \text{otherwise.} \end{cases}$$

For the case of $S = \text{CELL\_FACH}$, we have:

$$P(S = \text{CELL\_FACH}|A = \tau) = \begin{cases} 0, & \text{if } \tau \leq T_{\text{DCH}} \\ \frac{\tau - T_{\text{DCH}}}{\tau}, & \text{if } T_{\text{DCH}} < \tau \leq T_{\text{DCH}} + T_{\text{FACH}} \\ \frac{T_{\text{FACH}}}{\tau} & \text{if } \tau > T_{\text{DCH}} + T_{\text{FACH}}. \end{cases}$$

The probability for the CELL_DCH state $P(S = \text{CELL\_DCH}|A = \tau)$ does not differ from the Two State Model, i.e. Equation 2.3.

Average Frequency of State Transitions In contrast to the Two State Model, we have to consider a larger number of state transitions. These are the transitions from RRC_`Idle` to CELL_DCH, from CELL_DCH to CELL_FACH, from CELL_FACH to CELL_DCH, and from CELL_FACH to RRC_`Idle`. Other transitions do not occur. We first calculate the frequency of state transitions from CELL_DCH to CELL_FACH. This transition happens every time the inter-packet time A exceeds the timer T_{DCH}. Therefore, the derivation is the same as presented above:

$$E[f_{\text{CELL\_DCH}\rightarrow\text{CELL\_FACH}}] = \frac{1 - A(T_{\text{DCH}})}{E[A]},$$

$$E[f_{\text{CELL\_FACH}\rightarrow\text{RRC\_}\texttt{Idle}}] = \frac{1 - A(T_{\text{DCH}} + T_{\text{FACH}})}{E[A]}.$$

Furthermore, all state transitions from CELL_FACH to RRC_`Idle` correspond to a switch from RRC_`Idle` to CELL_DCH and therefore $E[f_{\text{RRC\_}\texttt{Idle}\rightarrow\text{CELL\_DCH}}] = E[f_{\text{CELL\_FACH}\rightarrow\text{RRC\_}\texttt{Idle}}]$. Finally, we calculate $E[f_{\text{CELL\_FACH}\rightarrow\text{CELL\_DCH}}]$. These state transitions occur, if $T_{\text{DCH}} < A \leq T_{\text{DCH}} + T_{\text{FACH}}$. Therefore, we have

$$E[f_{\text{CELL\_FACH}\rightarrow\text{CELL\_DCH}}] = \frac{A(T_{\text{DCH}} + T_{\text{FACH}}) - A(T_{\text{DCH}})}{E[A]}.$$

Other state transitions do not occur in our scenario, as shown in Figure 2.3a).

Modelling Signalling Intensity and Power Drain of the UE

We assume that every state transition involves signalling traffic. In order to quantify signalling load on an abstract level, we define the Signalling intensity (SI) of an application, i.e. of a given distribution for A, as the average number of state transitions required for the transmission of a single data packet.

$$SI = \frac{E[f_{ST}] \cdot T_{\text{Obs}}}{E[n_{\text{P}}]} = E[f_{ST}] \cdot E[A] \tag{2.5}$$

where $E[f_{ST}]$ is the sum of all state transitions. Consequently, $SI \in]0, 2]$ for the Two State Model since every packet can at most cause two state transitions, in the Three State Model $SI \in]0, 3]$ holds. This metric is intended to quantify the relation between transmitted data packets and the involved RRC state transitions, which all incur mobile network signalling. The metric can be extended to capture more details, e.g. the number and type of signalling messages exchanged for a specific state transition, as discussed in Section 2.2.1. Since we will use this metric for a more qualitative analysis of source traffic produced by smart-phone applications, we stick to the definition above allowing for an illustrative understanding of the numerical results.

Next, we model the PD of the UE due to the UMTS transmission unit. We assume three power levels PD_S, one for every state s and calculate the average power drain PD based on the state distribution, which in turn depends on the packet arrival process A. We obtain

$$PD = \sum_{s \in S} PD_s \cdot P(S = s) \tag{2.6}$$

with $S = \{\text{RRC\_}\texttt{Idle}, \text{CELL\_FACH}, \text{CELL\_DCH}\}$ for the Three State Model or $S = \{\text{RRC\_}\texttt{Idle}, \text{CELL\_DCH}\}$ if the Two State Model is considered. This is a user-centric metric and gives insights into how efficient the transmission process uses the battery.

2.3.2 Impact of Analytic Traffic Characteristics

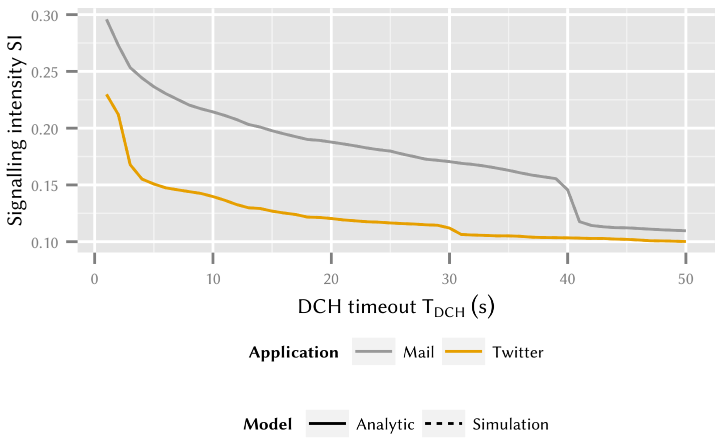
First, we validate our performance model by comparing the analytical results with simulations based on measured packet traces of two real smartphone applications. Then, we investigate the impact of traffic patterns on signalling load and power drain and derive high-level implications of the model.

Validation of Analytic Model

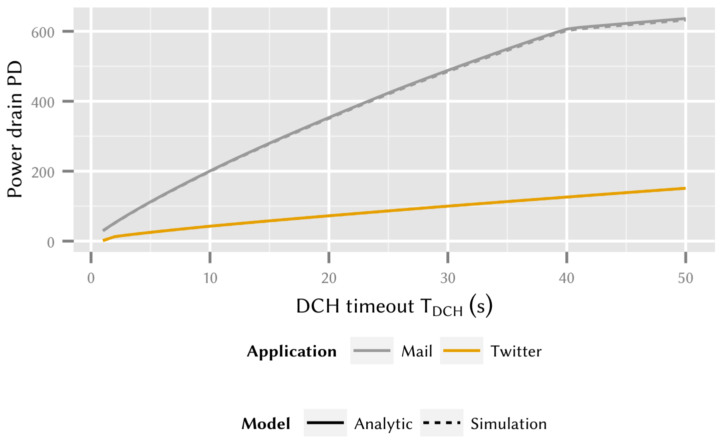
In order to assess the applicability of our performance model, we first have to check whether real-world application traces can be modelled as a renewal process, which was our main assumption for the model. We use the Lewis-Robinson-Test [45], which is a hypothesis test with null hypothesis H_0 that the tested process is a renewal process. To this end, we use exemplary measurements, obtained with the testbed introduced in Section 2.2.1, for two different types of applications: *Twitter* and *K9-Mail*. According to this test, the null hypothesis cannot be rejected for both of our packet traces at a significance level of $95\,\%$. Although this assumption may not be true for all applications, our results show that at least the considered applications can be modelled as a renewal process.

Next, we compare our analytical performance results with RRC protocol simulations using measured application and Transmission Control Protocol (TCP) traces which are described in more detail in Section 2.2.2. In order to produce analytical results that correspond to the real applications, we extract the empirical distributions of the inter-packet time A from the traces for both applications and use these distributions as input for Equation 2.3.1.

In Figure 2.15 we compare the accuracy of the results obtained by the presented method to the values obtained from simulations for the two measured applications and both considered metrics. We observe that the accuracy for both power drain PD and SI is very high. In Figure 2.15a the results for both the Mail and Twitter application obtained by the model completely align with the signalling intensity obtained by the simulation. The comparison of analytical results for the power drain to the simulation in Figure 2.15b leads to the same conclusions as for the signalling intensity.



(a) Signalling intensity



(b) Power drain

Figure 2.15: Comparison of the performance model with a 3G simulation for the Three State Scenario. Lines overlap due to a high goodness of fit.

Impact of Traffic Patterns on Signalling Intensity

First, we focus on the signalling intensity SI of traffic patterns and check the impact of the average inter-packet time $E[A]$ and the timer configuration. The signalling intensity SI, i.e., the average number of state transitions required for the transmission of a single packet, is an abstract measure for the signalling load produced by a specific traffic pattern.

Impact of the Average Inter-Packet Time $E[A]$ Some applications, for example those downloading or streaming of videos, send and receive large amounts of data within short time frames. In contrast, other applications, e.g. social network clients send and receive only small amounts of data every few minutes over the time span of some hours or days.

In this section we study the impact of average inter-packet times $E[A]$ and the burstiness of the traffic pattern, i.e., the coefficient of variation

$$c_A = \frac{\sqrt{\mathrm{Var}[A]}}{\mathrm{E}[A]}$$

on the signalling load. For that purpose, we use the simple Two State Model, set the timer $T_{\mathrm{DCH}} = 10\,\mathrm{s}$, consider only the first and the second moment of the inter-packet time A, and assume that A follows a log-normal distribution, where both moments can be varied independently.

In Figure 2.16, we vary the average inter-packet time $E[A]$ in six orders of magnitude and investigate the resulting signalling intensity SI for different coefficients of variation c_A. We observe that c_A has no impact on SI for very small inter-packet times $E[A] < 1 \times 10^{-1}$ s. Here, the UE stays in state CELL_DCH for the complete time since no inter-packet times $A > T_{\mathrm{DCH}}$ occur. In addition, the impact of c_A is small for very large values of $E[A] > 1 \times 10^{3}$ s. In this case, the UE switches to state CELL_DCH and back to state RRC_`Idle` for the transmission of every packet. Therefore, the signalling intensity SI approaches the value 2. For values in between these two extremes, the coefficient

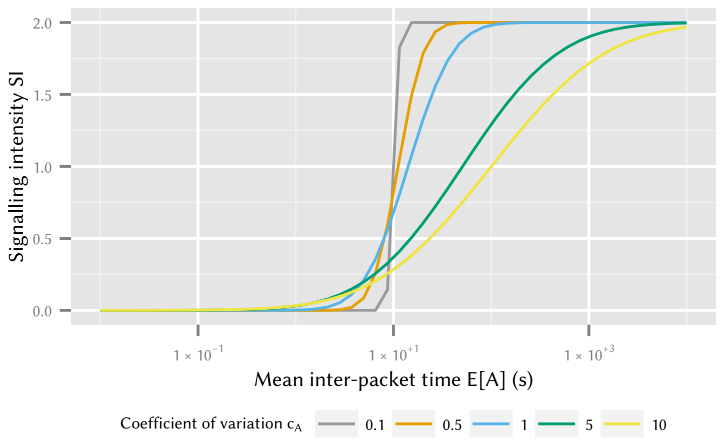


Figure 2.16: Signalling intensity SI for different traffic patterns considering the Two State Model with $T_{DCH} = 10\,\mathrm{s}$ for different traffic patterns.

of variation c_A has a considerable impact on the signalling intensity SI. More periodic traffic, i.e. smaller values of c_A, results in an increase of SI from 0 to 2 very sharp at the value $E[A] = T_{\mathrm{DCH}}$, while this increase is smoother for larger values of c_A. This is due to the fact that for nearly periodic traffic it is crucial whether the timer value T_{DCH} is smaller or larger than $E[A]$. For larger values of c_A this dependency is weaker.

Impact of the Coefficient of Variation of the Inter-Packet Time c_A Next, we focus on the impact of the timer value T_{DCH} with respect to the burstiness of the traffic. We use the same setting as before, but fix the average inter-packet time $E[A] = 4\,\mathrm{s}$. While there are differences in $E[A]$ amongst users in real world settings, measurement studies have revealed that across all users 95 % of the packets are received or transmitted within 4.5 s of the previous packet [46]. Therefore, the order of magnitude of $E[A] = 4\,\mathrm{s}$ is of practical relevance.

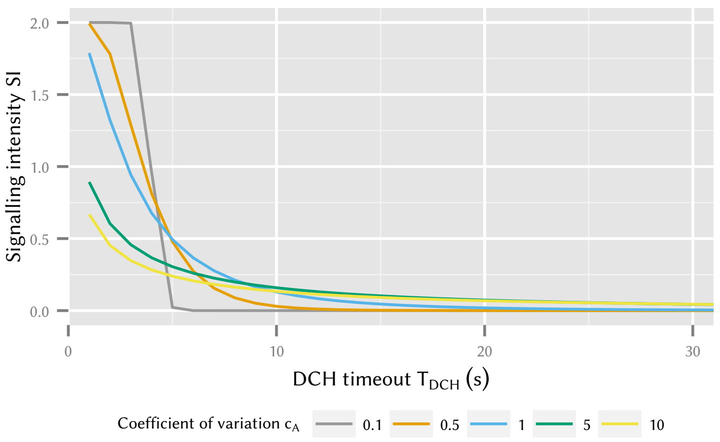


Figure 2.17: Signalling intensity SI for the Two State Model w.r.t. different timeout values T_{DCH} and coefficient of variations c_A.

The signalling intensity SI is shown in Figure 2.17 with respect to the timer value T_{DCH} and the burstiness c_A of the traffic pattern. Obviously, larger timers lead to less frequent state transitions and therefore to less signalling load. In addition we observe that the impact of the timer is crucial for nearly periodic traffic. If the average inter-packet time for nearly periodic traffic is larger than the timer, then every packet transmission involves a state transitions from RRC_`Idle` to CELL_DCH and a transition back. In contrast, no transitions are required if the average inter-packet time is shorter than the timer. With increasing values of c_A the impact of the timer is reduced. This means that for bursty traffic patterns the timer value is of less importance with respect to the generated signalling load.

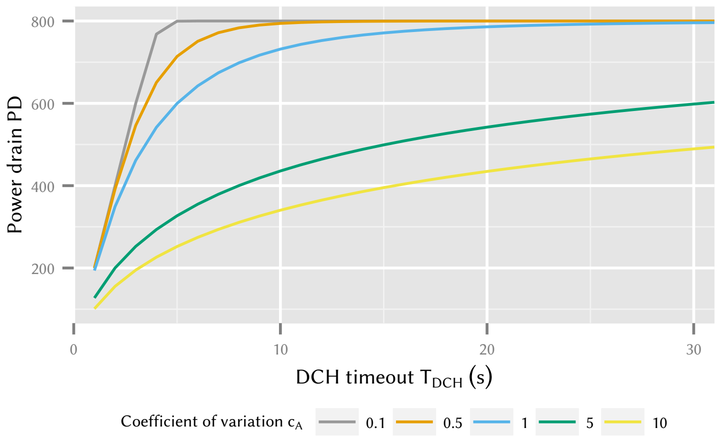


Figure 2.18: Power drain PD for the Two State Model w.r.t. different timeout values T_{DCH} and coefficient of variations c_A.

Impact of Traffic Patterns on Power Drain of the UE

In this section we study the impact of the traffic patterns on the power drain PD of the UE. This metric quantifies how resource-efficient specific traffic patterns and timer configurations are for the battery of the UE.

For the power drain in the different RRC states, we use the same radio network power drain used in Section 2.2.1. We investigate the impact of the average inter-packet time, the impact of the timer configuration and validate our model with simulations. In Section 2.3.2 we have seen that no state transitions occur for very small and very large average inter-packet times $E[A]$. This was due to the fact that for very small values the UE is continuously in state RRC_`Idle` and for large values it switches to state CELL_DCH for every packet. Thus, traffic patterns with very small and very large inter-packet times $E[A]$ have also no impact on the power drain of the UE regardless of the burstyness represented by the coefficient of variation c_A.

To study the impact of the timer configuration T_{DCH}, we use the same setting as for the signalling load: log-normal distribution of inter-packet time A, $E[A] = 4\,\text{s}$ in the Two State Model. The numerical values shown in Figure 2.18 indicate that longer timeouts lead to a higher power drain PD. This is reasonable since the UE stays longer in the power intensive CELL_DCH state in these cases. However, we observe that the burstiness of the traffic pattern has also a considerable impact on the power drain PD. For example, for $T_{\text{DCH}} = 15\,\text{s}$, the power drain is only $400\,\text{mW}$ for very bursty traffic with $c_A = 10$, while it is almost $800\,\text{mW}$ for less bursty traffic with a $c_A = 1$. The reason is that bursty traffic patterns send a lot of traffic during short periods when the UE is in state CELL_DCH anyway. During the following off-periods that UE can save power in RRC_`Idle` state. Hence, we conclude that longer timeouts and smaller coefficients of variation $c_A = 1$, i.e. more periodic and less bursty traffic, result in a higher power drain of the UE.

Tradeoff: Energy Consumption vs. Signalling Load

In Figure 2.19, we show the effect of network parameter optimisation using the timer T_{DCH} on traffic patterns with varying coefficient of variation.

We see that optimisations may decrease signalling by large amounts while only having very little impact on power drain for one specific kind of traffic. The same timer setting could increase the power drain for another kind of traffic while only offering little benefit with regard to the generated signalling intensity.

2.4 Lessons Learned

In this chapter we studied the impact of smartphone application traffic on mobile communication networks. We considered three stakeholders interacting in the mobile network. The *mobile network operator* is interested in preventing so called signalling storms, where network components performance is degraded due to high signalling load caused by applications generating network traffic

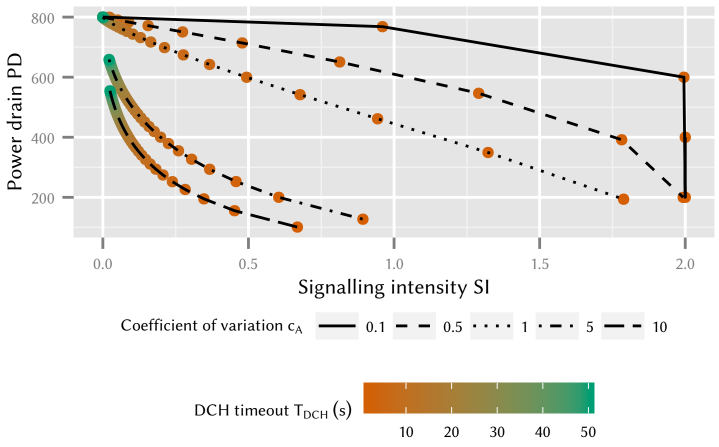


Figure 2.19: Trade-off between PD and SI for the Two State Model.

from users' UEs. The *hardware vendor* is interested in satisfying customers by providing a long battery lifetime for the UE, i.e. reducing power drain. The *application developer* is interested in increasing QoE for the applications user. Each of the stakeholders can influence the mobile network, by manipulating the parameters under its control. The network operator can manipulate RRC timers, increasing the time a smartphone stays connected to the network if no data is sent or received, decreasing the number of connections being established or severed and thus the signalling load in the network. The hardware vendor can implement proprietary RRC protocol extensions, skipping power intensive connection states in order to reduce power drain. The application developer can shorten update intervals, in order to provide more up to date events and increase QoE. However, each of the parameters under the control of the individual stakeholders influence the KPIs of the other stakeholders.

This chapter provides a two-pronged approach to analysing the impact of changes by individual stakeholders on the overall network.

First, we provided an algorithm to derive RRC state transitions from traffic measurements of already deployed or prototyped applications. While proprietary mechanisms exist to directly measure RRC state transitions, due to the high price they are usually out of reach for application developers, preventing them from evaluating the impact of their applications on the network. Based on this algorithm we analyse four popular smartphone applications, and find that while it is possible to find a viable tradeoff between signalling load and power drain for single applications, no such tradeoff exists if multiple applications operating in the network at the same time are considered. For example, for the considered *Twitter* application, increasing the network timer T_{DCH} from 10 s to 11 s would result in a decrease of signalling by 40 %, while only resulting in an increase of power drain of 6 %. However, if the *Aupeo* application is running in the same network optimised for the Twitter application, this change results in no reduction of signalling load and an increased power drain of 5 %.

Furthermore, we show that network timer optimisation, a practice where network operators manipulate RRC timers in order to reduce signalling load, incentivises users to enable proprietary fast dormancy algorithms, resulting in a net increase of signalling load. For example, if a network operator increases the T_{DCH} network timer from 4 s to 8 s, in order to reduce the signalling frequency caused by the Angry Birds application by 67 %, this results in an increased power drain at the user's UE of 341 %. If the user enables the fast dormancy option of the UE, the power drain is decreased by 27 %; however, this increases the signalling frequency above the original value before the reconfiguration of the network operator.

Second, we propose an analytical model to derive the KPIs from analytical or empirical traffic distributions, in order to evaluate the impact of applications that do not yet exist or classes of applications defined by a common traffic characteristics. Our results show that different access patterns have a considerable impact on the required resources of the mobile phone and the network. We identified bursty traffic patterns as particularly resource-efficient with respect to power drain and signalling load. In contrast, nearly periodic traffic is likely to cause signalling overload due to frequent connection re-establishments, especially when the connection timeout is slightly below the inter-packet time. This can be observed on the example of a T_{DCH} timer of 10 s. Here, the coefficient of variation has no impact on the signalling load for very small inter-packet times $E[A] < 1 \times 10^{-1}$ s or very large inter-packet times $E[A] > 1 \times 10^{3}$ s. For example, for a mean inter-arrival time of $E[A] = 11.5$ seconds, an increase of coefficient of variation from 0.5 to 5.0 can decrease the signalling load by $53\,\%$.

Concluding from this chapter, we see that in mobile networks many different players, metrics, and tradeoffs exist. We highlighted one example of such a tradeoff, i.e. signalling load vs. power drain and discussed the influence of the current optimisation parameters, the network timers, on another. However, many additional tradeoffs exist. For example, the mobile operator has to balance the use of radio resources with the number of generated signalling frequencies. Furthermore, application providers seek to improve the user experience which usually result in a higher frequency of network polls, creating additional signalling traffic. The high number of tradeoffs and involved actors in this optimisation problem indicate that the current optimisation technique used by operators is no longer sufficient.

Approaches like *Economic Traffic Management* [47] or *Design for Tussle* [48] could be applied to find an acceptable tradeoff for all parties. In Economic Traffic Management all participating entities share information in order to enable collaboration. This collaboration allows for a joint optimisation of the tradeoff. Design for Tussle aims to resolve tussles at run time, instead instead of design time. This prevents the case that one actor has full control over the optimisation problem, which would likely result in the actor choosing a tradeoff only in its favour, ignoring all other participants. One example of an actor providing information for another in order to optimise the total system would be a UE vendor providing interfaces for application developers to use when sending data. These interfaces would schedule data to be transmitted in such a way that signalling load and power drain would be reduced, if the application's requirements allow for it. Until such interfaces exist, application developers could take the effect of the traffic their applications produce both on the UE and the network into account, for example using the algorithms proposed in this chapter.

3 Application Behaviour in Mobile Networks

While the previous chapter focussed on the network and the impact of application traffic on hardware vendors, network operators and users, this chapter shifts focus on the application behaviour themselves. The Internet supports a multitude of different applications, including video streaming services, online gaming, file storage services, cloud office solutions, etcetera. In contrast to applications of earlier generations, todays' services are not standardised or under control by network operators but rather the result of free enterprise and entrepreneurship. The interaction of such applications with the network and other players, e.g. cloud providers, are usually not considered by the application developers. Deploying such applications can impact other stakeholders, e.g. *Signalling Storms* interfering with the operation of mobile networks as discussed in Section 2.2. As this is of no consequence to application developers, they have no incentive to investigate the impact of their applications on other stakeholders.

In this chapter, we study the impact of two of the most prominent application types: *Video Streaming* and *File Synchronisation*, chosen due to their impact on global traffic and frequency of use. While in the last chapter we were able to rely on standard documents to model the systems under study, this option is no longer available when considering modern applications. Thus, we perform measurements or take studies of other researchers into account in order to obtain knowledge of both the systems under study as well as the related stakeholders.

Similar to Chapter 2, we identify a set of involved stakeholders and their respective key performance indicators and derive corresponding metrics, as

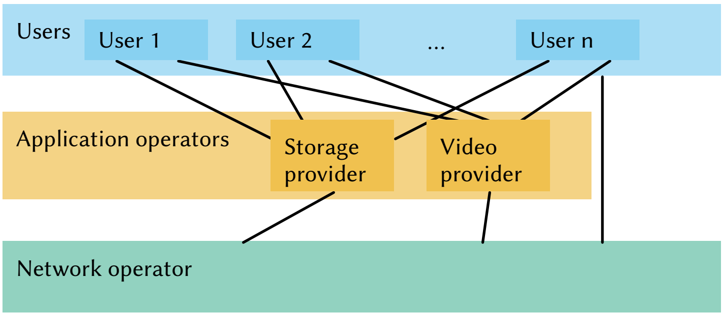


Figure 3.1: Interactions between considered stakeholders in the application scenarios.

shown in Figure 3.1. First, we consider the *application provider*. In the case of the Video Streaming scenario, this role is the part of the video provider. We consider the video provider to be interested in two performance indicators: *a*) user satisfaction, realised by a QoE metric, *b*) cost reduction in compute and network infrastructure. In the case of the File Synchronisation scenario, we consider the application operator to be interested in user satisfaction, again realised as a QoE metric, the mean time to synchronisation. Second, we consider the *network operator*. As in the last chapter, they are interested in reducing load on the network infrastructure, in order to prevent cases similar to Signalling Storms. We measure this key performance indicator by considering the number of connections to the mobile network required to complete the synchronisation operation. Similarly to Chapter 2, we assume that the *user* is interested in both a high QoE as well long battery life for the used device, sometimes also considered a QoE metric [37], represented by the energy consumption metric.

The contribution of this chapter is threefold:

a) We provide models for video transmission mechanisms and perform a performance evaluation and tradeoff analysis considering metrics relevant to network operators, users, and video providers.

b) We provide a QoE model for video streaming allowing for the analysis of heterogeneous user profiles and use this model in order to evaluate the impact of user preference on video streaming scenarios.

c) We propose a model for cloud file synchronisation and evaluate a set of scheduling mechanisms regarding impact of considered metrics for all stakeholders.

The content from this chapter has been published in [9, 11, 14]. In Section 3.1 we discuss the current state of the art regarding video transmission mechanisms and QoE studies. Then, in Section 3.2 we discuss tradeoffs between different video transmission mechanisms, regarding the considered metrics. In Section 3.3 we study the impact of user profiles on the QoE experienced during video streaming. In Section 3.4 we consider the impact of different file synchronisation scheduling algorithms on the relevant stakeholders. Finally, we discuss lessons learned in Section 3.5.

3.1 Background and Related Work

This section first introduces the current state of the art of video transmission mechanisms in the network in Section 3.1.1. Then, we shift focus to the user in Section 3.1.2. We discuss related work regarding QoE for video playback including QoE modeling approaches, user profiles, and QoE management mechanisms.

3.1.1 Video Streaming Mechanisms

In order to transfer video content from the content providers to the users over the Internet, multiple solutions exist [49]. The most basic approach, *Download*, obtains the complete video at once, playing back any available content as required. Due to the nature of *Live* video transmissions it is only possible to send the currently available content. Furthermore, introducing delay into the live-stream should be avoided as it reduces the timeliness of the video. There exist different approaches for *Streaming* video content to a user. In server-based solutions, the streaming server controls the transmission of content. One example of such a server based approach is the Real-Time Streaming Protocol (RTSP) which was widely discussed as a standardised solution for mobile video streaming [50].

In the more recent past, client-based approaches were discussed. Here the client side controls the download and playback of content. The authors of [51] study the QoE of HTTP Adaptive Streaming (HAS) approaches in LTE networks. They highlight the differences to existing server-side approaches and suggest the study of cross-layer optimisation approaches in order to improve the QoE. One approach to deliver HAS is Dynamic Adaptive Streaming over HTTP (DASH), which enables video streaming over Hyper Text Transfer Protocol (HTTP) [52].

The increasing popularity of video streaming has driven intensive research activities on how to optimise the video delivery to the end user concerning QoE. In particular, HTTP streaming is deployed by large video service delivery platforms, e.g. YouTube or Netflix and represents the major video delivery solution, especially for video-on-demand.

In HTTP video streaming, video data is transmitted to the client via HTTP and stored in an application buffer. After the download of a sufficient amount of data q, which is in the order of a few video seconds, e.g. for YouTube, the video play out starts at the client. As soon as the video buffer falls below a certain threshold p, the video stalls [53]. In the remainder of this work, we refer to this threshold policy as pq-policy.

3.1.2 Video Quality of Experience for HTTP Adaptive Streaming

The goal of HAS is to adapt the video to the current network conditions. The video adaptation may be realised by changing the frame rate, resolution, or quantisation of the video. Although the adaptation results in lower quality, the major benefits compared to classical HTTP video streaming is the reduction of stalling events. The authors of [54] survey QoE for HTTP adaptive streaming and give an overview of recent developments. Besides improved quality adaptation mechanisms [55], other approaches aim for example at optimising the segmentation of the videos [56].

Subjective studies showed that users prefer initial delays instead of stalling events [57]. An analytical framework for the dimensioning of appropriate video buffers for TCP streaming shows that the initial buffering delay and the size of the buffer should be as small as possible, yet large enough to avoid buffer underflows [58]. A concrete approach [59] determines the optimal, i.e. minimal, initial delay at the client. During this time, the video buffer is filled such that no stalling occurs. Two buffer size adaptation policies are proposed by [60] which are evaluated by means of a fluid model in terms of freezing probability.

The authors of [61] evaluate the impact of network dynamics and QoS provision on users video quality. An analytical framework models the playback buffer at the receiver as a $GI/GI/1$ queue, however no pq-policy is considered. Further, video quality is considered in terms of the start-up delay or fluency of video playback. Based on that, adaptive play-out buffer management schemes are proposed.

Considering both the video content as well as the available resources by using a proxy has been suggested to improve the users QoE [62] for HAS. In [63] the authors suggest the use of a caching strategy, downloading video content according to a user's viewing history and network conditions.

So far, no queueing system with pq-policy is applied to analyse QoE for HTTP video streaming and to dimension video buffers accordingly. In queueing theory, the related threshold policy is denoted as N-policy introduced by [64] with $p = 0$ and $q = N$; the server stops whenever the system becomes empty and resumes service when the number of waiting customers in the system, i.e. the video buffer in our case, reaches a threshold value N. Section 3.3.1 will show that in contrast to the transient phase, in the steady state q has no influence on the performance.

Various researchers analysed the N-policy. In [65] the stationary joint distribution of queue length and the server's status for the $GI/M/1$ is derived. The authors of [66] obtain the steady state probability distribution of the number of customers in a finite system for the $M/GI/1$ system with N-policy. A transient solution of the $M/M/1$ queue under pq-policy is derived by [67].

Results from queueing theory may be applied to dimensioning the video buffer for HTTP streaming in order to optimise QoE. However, the approaches mentioned above are either considering QoS parameters only or they apply QoE models based on MOSs of subjects. However, differences in how QoE degradations are observed by individual users are not considered. In Section 3.3 we propose an analytical model which allows to investigate individual user profiles based on a parametrised QoE model.

Most user studies on HTTP video streaming quantify and report QoE in terms of MOS, e.g. [53]. However, there is a diversity in user perception which is eliminated by the process of averaging subjective ratings. A relationship between the MOS and the second moment of the user ratings is formulated as Standard deviation of Opinion Scores (SOS) hypothesis and a standard deviation for particular MOS values is observed up to 0.8 for video QoE [68]. Thus, user perceptions may fluctuate between good and poor quality under the same conditions. The authors observe different user types, denoted as *hectic*, *regular*, *insensitive* depending on their sensitivity to QoE degradations.

Various resource management mechanisms to improve QoE for YouTube have been proposed in the literature, e.g. in Wi-Fi mesh networks [69] or using Software-Defined Networking (SDN) [70]. SDN enhances the interaction between networks and applications and allows a more dynamic and demand-based allocation of network resources which is demonstrated for YouTube video streaming. To overcome resource limitations in the content delivery infrastructure, [71] proposes client-based local caching, Peer to Peer (P2P)-based distribution, and proxy caching which reduces network traffic significantly and can therefore avoid QoE degradations.

3.2 Trade-Offs for Multiple Stakeholders in LTE Video Transmission

The delivery of video content in a mobile scenario is one of the major use cases for the LTE mobile communication technology. While the use of LTE affords sufficient bandwidth to enable video playback in high definition, it also introduces new challenges for all stakeholders. Similarly to the applications discussed in Chapter 2, signalling traffic induced by the video transmission may pose a challenge for mobile network operators, and power drain remains an open issue for hardware vendors. Additionally, mobile playback can cause new problems for video providers. Users watching video on the go may be prone to higher rates of interrupted video playback, either due to insufficient network coverage or social interactions. If the video transmission mechanism has transmitted too much video content in advance, this transmitted data is “wasted”, from the perspective of the video provider, as this consumes both network and computation resources but results in no benefit for the customer.

In Section 3.2.1 we present models for both video playback and the mobile network. Then, in Section 3.2.2 we perform a simulative study using the proposed model in order to evaluate the performance of the studied video transmission mechanisms.

3.2.1 Model of LTE Video Transmission

In this section, we present modeling assumptions used in the remainder of this section. Then, we propose a system model for both video transmission mechanisms and the considered LTE network. Finally, we introduce performance metrics for each of the considered stakeholders.

Transmission Model Assumptions

Maintaining a high QoE for their viewers is an important goal for operators of video platforms. The authors of [72] find that the QoE is mainly influenced

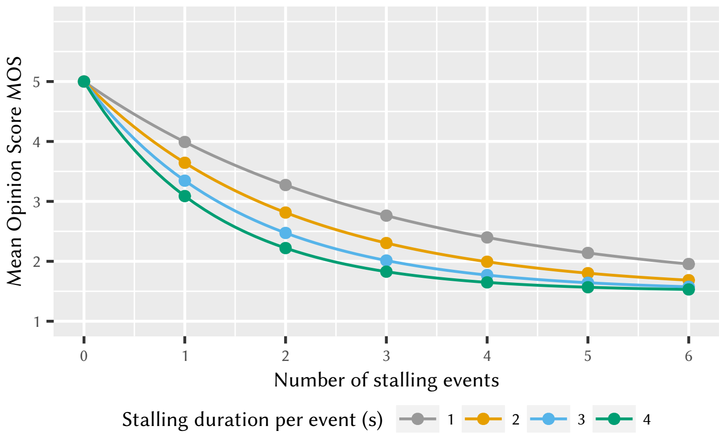


Figure 3.2: Influence of number of stalling events and stalling length on QoE [72].

by the number of stalling events and the stalling event duration. As shown in Figure 3.2, the QoE model, where 5 is the highest possible MOS and 1 the lowest, rapidly decreases if the number or duration of stalling events increases. The provided QoE model between stalling and QoE shows that stalling significantly worsens QoE. Thus, an operator has to avoid stalling at any cost. As a consequence, we only consider scenarios where no stalling occur, i.e. the delivery bandwidth is larger than the minimum video bit rate to ensure a smooth video play-out. Furthermore, in Section 3.3 we will study the impact of available network load and user preference on QoE.

Furthermore, we assume that all videos are played back with a constant bit rate b_R. Thus, each second of the video, independently of its content, requires the same number of bits.

We consider video transmission between a server and a user equipped with an LTE enabled smartphone. The available bandwidth of a UE depends on many factors, e.g. location, number of users in the cell, activity of other users, and line of sight. To simplify the evaluation scenarios we assume that a constant maximum bandwidth b_W is provided to the user. We assume that the bottleneck of the connection is the air interface, thus the full available bandwidth b_W is used for the video download, which prevents stalling.

Although the assumptions of constant bit rate and bandwidth do not hold in a real environment, the purpose of considering such assumptions is twofold. First, they are useful in order to analyse the performance of the discussed mechanisms in optimal conditions without any other effects that could disturb the results and, second, they can serve as a baseline for comparison with fitted random variables for both bandwidth and bit rate.

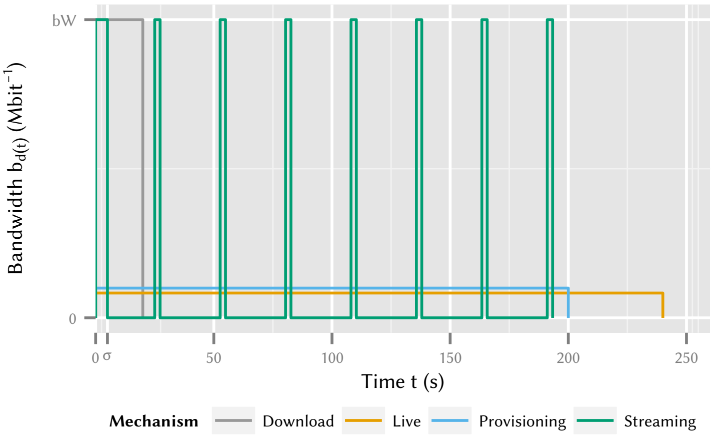
Considered Video Traffic Model

In our study we focus on four transmission mechanisms which are currently in use. Figure 3.3b shows the consumed bandwidth $b_d(t)$ and the available seconds of video for playback $t_u(t)$ for a video for all considered transmission mechanisms at all points of time t. Furthermore, for the remainder of this chapter, we refer to the amount of video in seconds already played back at a point in time t as $t_p(t)$.

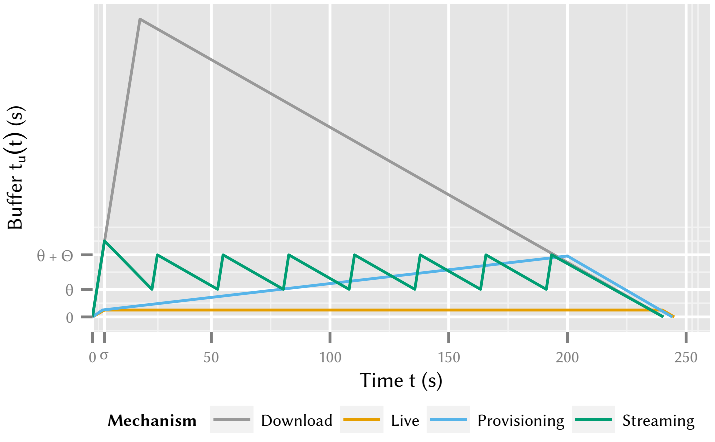
a) **Download** The *Download* mechanism can be used if a user wants to watch a pre-encoded video. Thus, the complete video is ready to be transmitted as soon as the user starts the transmission. The required time of the download is only bounded by the bandwidth available in the network.

b) **Live** Video watched during *Live* transmissions is encoded as it is recorded. Thus, the bandwidth used to transmit the video is always limited to the video bit rate b_R.

c) **Provisioning** In [73] the authors show the influence of the video demand, i.e. the ratio of available bandwidth and required video bandwidth, on the stalling frequency. In order to reduce stalling, the bandwidth used



(a) Bandwidth consumption b_W



(b) Remaining playback buffer $t_u(t)$

Figure 3.3: Behaviour of transmission mechanisms. Different playback end times occur due to different playback starts.

to download the video should be provisioned so that the available bandwidth exceeds the video bandwidth by a high enough factor. In the *Provisioning* mechanism, the download bandwidth is chosen so no stalling occurs. In order to reduce stalling and improve the QoE of a video, the available bandwidth should be at least 120 % of the video bit rate b_R [74].

d) **Streaming** For the *Streaming* mechanism the complete video is encoded in advance, allowing for the full bandwidth of the UE being used for download. The video is downloaded with full bandwidth for a *pre-buffering time* σ in order to guarantee a stalling-free start of the playback. After σ s the download stops and the playback begins. The download is only resumed if the available seconds of video for playback are below a *stop threshold* θ. The download continues until the buffer contains a *threshold size* Θ, resulting in a total buffer length of $\theta + \Theta$ s. This is repeated until the download is completed.

A video provider will also consider its bandwidth to be a resource to be conserved. However, when comparing the bandwidth available in LTE with that of a wired network, we can assume the air interface to be the bottleneck. Furthermore, not considering bandwidth as an optimisation target of the video provider simplifies the study as it removes one additional metric.

LTE Network Model for Video Transmission

In order to quantify the energy consumption during wireless transmission, we model the LTE RRC behaviour defined in [28]. To reduce the energy consumption, the concept of Discontinuous Reception (DRX) has been introduced in [75]. The authors of [76] provide measurements of important RRC and DRX parameters which are used in the following model and are reproduced in Table 3.1.

The RRC protocol for LTE consists of two states, as shown in Figure 3.4. In RRC_`Idle` state, the UE is in DRX mode. Here, the UE monitors the Physical Downlink Control Channel (PDCCH) for $T_{\text{ON}}^{\texttt{Idle}}$ in each DRX interval of duration $T_{\text{DRX}}^{\texttt{Idle}}$. The time of a promotion to the RRC_`Connected` state is given by

Full name	Symbol	Measured value
`RRC_Connected` *On* duration timer	T_{ON}	1 ms
DRX inactivity timer	T_{I}	100 ms
`Short` DRX duration timer	T_{S}	20 ms
`Long` DRX duration timer	T_{L}	40 ms
`RRC_Connected` timeout	$T_{\mathtt{Idle}}$	11.576 s
`RRC_Idle` *On* duration timer	$T_{\mathrm{ON}}^{\mathtt{Idle}}$	43 ms
`RRC_Idle` DRX duration timer	$T_{\mathrm{DRX}}^{\mathtt{Idle}}$	1.28 s
Promotion delay	D_P	260 ms

Table 3.1: Considered RRC and DRX parameters [76].

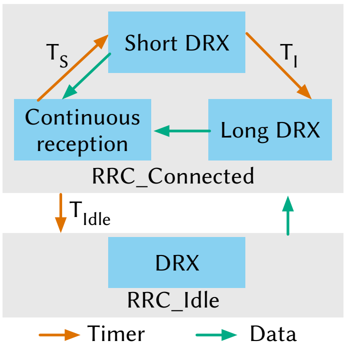


Figure 3.4: Considered LTE RRC model.

the promotion delay D_P and occurs as soon as a packet is sent or received. If a packet is sent or received while in RRC_`Connected`, including the initial packet which triggered the promotion to RRC_`Connected`, the timers T_{I} and $T_{\texttt{Idle}}$ are started. Until the T_{I} timer expires, the UE is in Continuous Reception (CRX) mode. After the $T_{\texttt{Idle}}$ timer expires, the UE demotes to RRC_`Idle`. Upon expiration of the T_{I} timer, the UE enters `Short` DRX. Here, the T_{S} timer is started and the UE monitors PDCCH for T_{ON}. If a packet is sent or received while in `Short` DRX, CRX begins and the T_{S} timer is disabled. Once the T_{S} timer expires, `Long` DRX is entered and T_{L} is started, again the UE monitors PDCCH for T_{ON}. This is repeated until a packet is sent or received and the CRX state is entered or until the $T_{\texttt{Idle}}$ timer expires and RRC_`Idle` is entered.

We give the download bandwidth at any time t as $b_d(t)$, as shown in Figure 3.3a. Furthermore, we denote the length of the video already downloaded at any time t as

$$t_d(t) = \frac{1}{b_R} \int_{\tau=0}^{t} b_d(\tau) d\tau. \tag{3.1}$$

Evaluation Metrics for Smartphone Energy Consumption, Wasted Traffic, and Connection Count

Each of the stakeholders is interested in different KPIs, from which we derive a set of metrics considered during the performance evaluation of the network and video playback model.

Energy Consumption We calculate the power drain of the UE due to wireless transmission at any given moment using the UE's current state and the bandwidth in use and aggregate it to the UE's energy consumption during the duration of the video playback. We only consider the energy consumption due to wireless transmission, as it is an offset to the energy consumption caused by the playback of the video. The video playback itself is unaffected by the choice of transmission mechanism. Thus, the selected transmission mechanism only influences the energy consumption of the wireless transmission. In [76], the au-

Description	Power drain
`RRC_Idle` (base)	11.4 mW
DRX during `RRC_Idle`	594.3 mW
Promotion	1210.7 mW
`RRC_Connected` (base)	1060.0 mW
DRX during `Short` DRX	1680.2 mW
DRX during `Long` DRX	1680.1 mW
α	$51.97\,\mathrm{mW\,Mbit^{-1}}$
β	1288.04 mW

Table 3.2: Power consumption per system state [76].

thors provided measurements for the energy consumption of each state, reproduced in Table 3.2, if the UE is receiving no data. Furthermore, an approximation of the power drain at time t if a download occurs is given as

$$P(t) = \alpha \cdot b_d(t) + \beta.$$

In order to compute the overall energy consumption E during the transmission and playback of the video, we aggregate the power consumed in each state in which the UE is not receiving and the power consumed during receiving while considering the used bandwidth at each moment.

Wasted Traffic If a user stops watching a video currently being downloaded before its end, this leads to *wasted traffic* as the data has been already prebuffered at the UE. This metric impacts the video provider, but is influenced by the user aborting the video. This decision can not be influenced by the video provider, thus a user model has to be assumed by the video provider in order to provide a performance analysis of the different video delivery mechanisms.

Considering that transmitting data to a smartphone consumes both compute as well as network resources, a transmission mechanism should attempt to re-

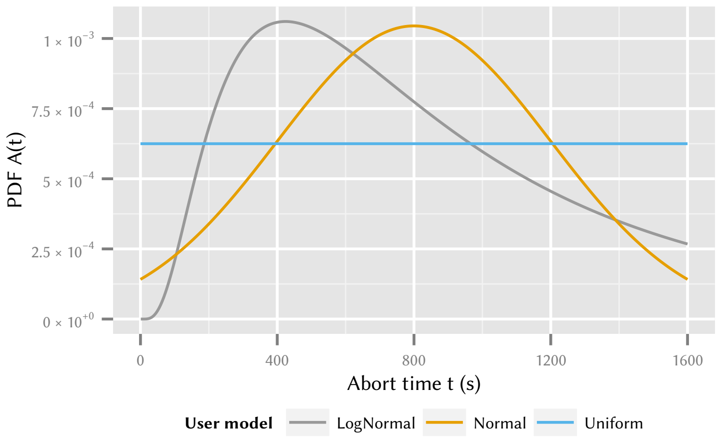


Figure 3.5: Considered user model abort distributions.

duce the amount of video which has been transmitted but is not yet watched at any time t as

$$t_u(t) = t_d(t) - t_p(t).$$

If the user stops the playback according to a random variable A with Probability Density Function (PDF) a, we can give the wasted traffic W as the expected value of t_u under A.

$$W = E\left[t_u\right] = \int_{t=0}^{\infty} a(t)t_u(t)dt. \tag{3.2}$$

High values of W indicate that server and network resources are used for traffic which is not watched by the user.

We consider three types of user behaviour, shown in Figure 3.5, each modelled by a random variable describing the abort time, i.e. the time when the user stops watching a video. First we consider a *uniform* distributed user abort model, where the user can abort the video at any time. Due to the uniform distribution of the abort time and the length of the video, i.e. 1600 s, the mean time of stop occurs at 800 s.

Second, we consider a type of user who watches a part of the video before deciding if the video is stopped. After the main part of the video has been watched, the user is again more likely to abort. To model this kind of behaviour we use a *truncated normal distribution* over the playtime of the video, assuming a symmetry of the abort density at the half-way point of the video. We use the same mean and specify a standard deviation of 400 s.

Finally, we assume that the user is more likely to abort the video at the beginning. We model this user behaviour using a *truncated log-normal distribution* with the same mean and a standard deviation of 0.8 s for the normal distribution at the basis of the log-normal distribution.

Note that the wasted traffic W is influenced by the user abort model, due to the fact that wasted traffic only occurs if a user aborts a video. Even though this may only affect a subset of all watched videos, it still consumes unnecessary resources and should be considered by the video provider. However, the download of videos always consumes energy and the largest amount of energy is consumed if the user does not abort the video. Thus, we optimise for the worst case energy consumption. Any other optimisation target would offer incentives to users to abort watching the video early, resulting in additional wasted traffic for the provider.

Connection Count A large portion of signalling messages caused by data transmission are generated if the UE switches between connected and disconnected state [28]. Thus, counting the number of connections C triggered provides an easy way to quantify the generated signalling during the video transmission.

3.2.2 Evaluation of Energy Consumption, Wasted Traffic, and Connection Count

In this section we study the metrics introduced in Section 3.2.1 on the different transmission mechanisms.

First, we study the impact of the considered transmission mechanisms on the energy consumption and the wasted traffic. Then, we consider the impact of the connection count for the Streaming mechanisms and varying values of the parameters *stop threshold* θ and *threshold size* Θ in more detail.

We consider a video of $l = 1600\,\mathrm{s}$ length which is viewed on a UE with LTE access. The median of available downlink throughput in current LTE networks is $b_W = 12.74\,\mathrm{Mbit\,s^{-1}}$ [76]. A wide set of video bit rates between $1\,\mathrm{Mbit\,s^{-1}}$ and $50\,\mathrm{Mbit\,s^{-1}}$ is in use[1]. In order to prevent stalling, we consider bit rates between $1\,\mathrm{Mbit\,s^{-1}}$ to $10\,\mathrm{Mbit\,s^{-1}}$, staying below the available network bandwidth. For the Streaming mechanism, a stop threshold of $\theta = 4\,\mathrm{s}$ and a threshold size of $\Theta = 32\,\mathrm{s}$ were selected. Furthermore, we specify a pre-buffering duration of $\sigma = 5\,\mathrm{s}$.

We conduct our study using deterministic discrete event simulation which uses no random variables. The wasted traffic is obtained analytically using the abort behaviour model. Thus, all results are exact under the previously stated assumptions.

Energy Consumption

First, we study the influence of both video bit rate b_R as well as the selected download mechanism on energy consumption in Figure 3.6.

We consider the Download mechanism and observe that it consumes the least amount of energy. Here the video is downloaded with full bandwidth, as seen in Figure 3.3, resulting in a very short energy intensive download phase and a longer energy playback phase which consumes less energy. For the Live mecha-

[1]`https://support.google.com/youtube/answer/1722171?hl=en`, Accessed: November, 21st 2015

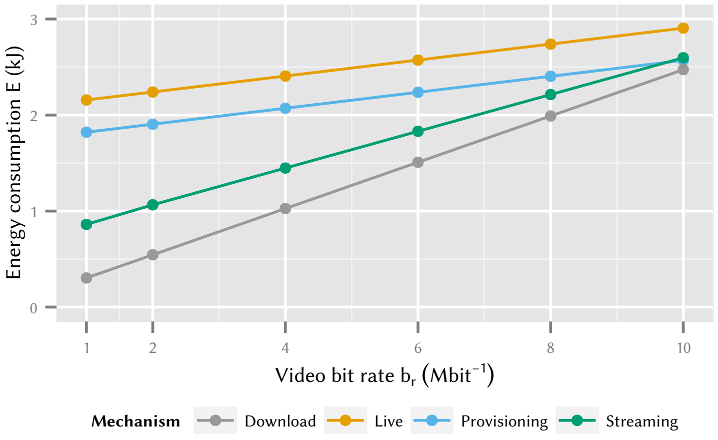


Figure 3.6: Influence of bit rate b_R and selected download mechanism on energy consumption E.

nism we observe the opposite, i.e. the highest energy consumption for all bandwidths. If this mechanism is used, the used bandwidth equals the video bit rate b_R. Thus, the download requires the same amount of time as the playback, resulting in the highest possible energy consumption E. The Provisioning method uses a higher bandwidth, thus reducing the overall download time. This reduced download time decreases the energy consumption E when compared to the Live mechanism, even though the bandwidth used for downloading is increased to $120\,\%$. For the Streaming mechanism we observe an energy consumption E slightly higher than the Download mechanism. As the bit rate b_R of the video increases, the energy consumption E increases as well. This is due to the fact that higher video bit rates b_R require larger downloads. For video bit rates b_R approaching the available bandwidth the Streaming mechanism degenerates to the Live mechanism, as no pre-buffering is possible. We conclude that the Download and Streaming mechanisms outperform Live and Provisioning with regard

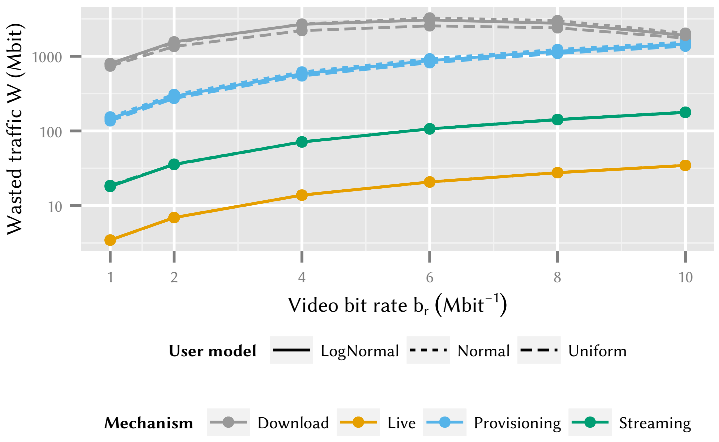


Figure 3.7: Influence of bit rate b_R, download mechanism and user model on the mean wasted traffic W. Download, Live, and Provisioning mechanisms result in equal connection counts.

to energy consumption.

Wasted Traffic

Next, we consider the wasted traffic W as a metric of the transmission mechanism quality. If a user completely watches a video, no traffic is wasted, as all data downloaded is used during playback. Thus, we consider only the cases where a user stops the playback before the video is finished.

In Figure 3.7 we study the mean wasted traffic W for different video bit rates b_R. We consider the different transmission mechanisms introduced in Section 3.3 as well as the previously introduced user models. We observe that the choice of user model has no significant impact on the mean wasted traffic W. For the Download mechanism, the amount of mean wasted traffic W increases up to a video bit rate b_R of $6\,\mathrm{Mbit\,s^{-1}}$, then the mean wasted traffic W decreases as only video data which has been pre-buffered can be lost if the user aborts the video. As we assume an available bandwidth of $12.74\,\mathrm{Mbit\,s^{-1}}$, the bandwidth available for pre-buffering decreases as the bit rate b_R increases, resulting in lower amounts of mean wasted traffic W for high video bit rates b_R. For the Live mechanism, we see that the wasted traffic for all user models is very low, but wasted traffic exists. This is due to the traffic already sent by the server while the UE is still waiting for promotion from RRC_Idle to RRC_Connected, i.e. a short pre-buffering phase exists. As the bandwidth increases with the video bit rate b_R, the mean wasted traffic W increases as well. Next, we consider the Provisioning approach and see an increase of mean wasted traffic W as the video bit rate b_R increases, due to the fact that the bandwidth used for continuous download is a factor of the video bit rate b_R. A higher video bit rate b_R results in the download of the video being completed earlier, which leads to more mean wasted traffic W. Similar results can be seen for the Streaming mechanism, which results in more mean wasted traffic W than the Live mechanism, but significantly less traffic than the Provisioning mechanism. This is due to the fact that if the user aborts, at least the amount of video given by the *stop threshold* θ and at most the complete buffer, given by the *stop threshold* and the *threshold size* are lost. We have observed that the choice of user model results in no qualitative changes in mean wasted traffic W. As described in the last paragraph, the Download and Streaming mechanisms provide best results with regard to energy consumption. However with regard to wasted traffic, the Live and Streaming mechanisms are most suited. Thus, the Streaming mechanism seems to be a good compromise. The network operator can select a tradeoff between energy consumption and mean wasted traffic W as discussed in the next section. From now on, we only consider the uniformly distributed user model.

Connection Count

The Internet Service Provider (ISP) is interested in reducing the number of connections occurring during video transmission. Thus, we quantify the impact of the selected video transmission mechanism on the connection count C, which

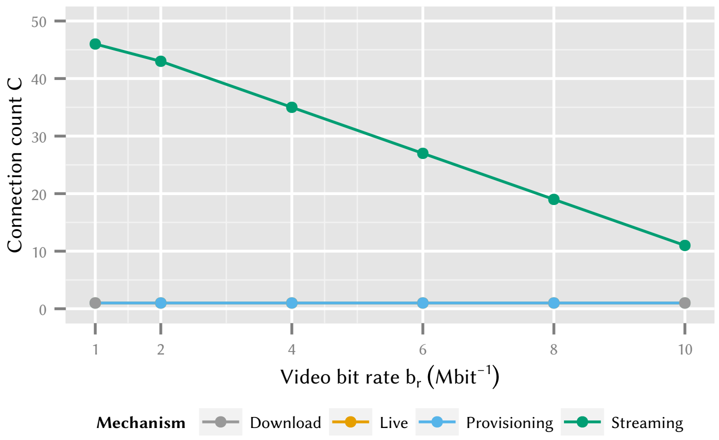


Figure 3.8: Influence of bit rate b_R and selected download mechanism on the connection count C.

directly correlates with the occurring signalling.

In Figure 3.8 we study the impact of the different transmission mechanisms on the number of connections per transmission and thus the amount of generated signalling. We observe that for the transmission mechanisms download, live, provisioning the connection count C is constantly one, independent of the selected bit rate b_R. This is due to the fact that in these transmission mechanisms the video is transmitted in one chunk. For streaming, the connection count C decreases as the video bit rate b_R increases. Here, a connection occurs each time the buffer is refilled. For larger bit rates b_R, refilling the buffer requires a longer transmission. As the maximum time of video transmission is upper bounded by the video length, longer buffering phases result in a smaller total amount of buffering phases and thus in a smaller connection count C per video transmission.

Next, we consider the impact of the stop threshold θ and buffer size Θ on the

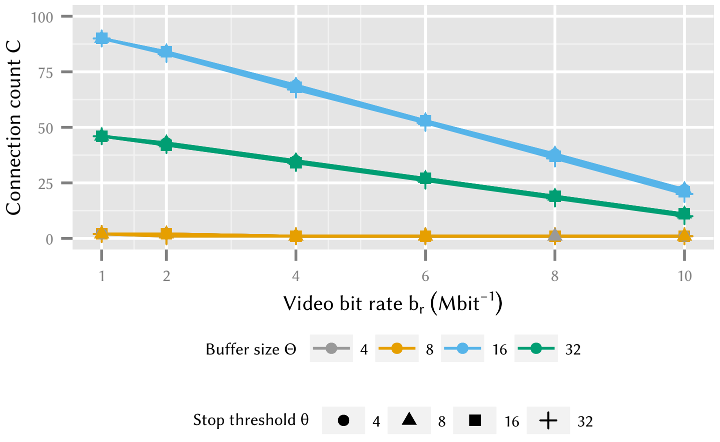


Figure 3.9: Influence of bit rate b_R and selected parameters on the connection count C for the streaming mechanism. Lines for different stop thresholds θ overlap.

connection count C caused by the Streaming mechanism. In Figure 3.9 we observe that while the buffer size has a significant impact on the connection count C during a video transmission, the lower buffer threshold has almost no impact. For buffer sizes of $4\,\mathrm{s}$ to $8\,\mathrm{s}$, no new connections are started, i.e. no signalling occurs. This is due to the fact that the connection timeout in UE is configured as $11.576\,\mathrm{s}$, as discussed in Section 3.2.1. Thus, for this low buffer sizes the UE does not disconnect from the network. Furthermore, we observe that as the buffer size increases, the number connection count C decreases. Refilling larger buffers requires, similar to larger bit rates b_R, longer transmission times. Thus, due to the total upper bound on the transmission time, less download phases can occur during the transmission.

3.2.3 Tradeoff Considerations for Participating Stakeholders

As shown in Section 3.2.2, the different video transmission mechanisms influence the performance of all considered metrics . In this section, we discuss the relationship of the metrics to each other.

First, we provide a high-level overview over available tradeoffs when selecting one of the introduced transmission mechanisms. Then, we study specific tradeoffs for the Streaming mechanisms regarding the considered metrics in more detail.

Impact of Selected Transmission Mechanism

Based on the observations regarding the metrics energy consumption, wasted traffic and signalling, we quantify the impact of the different transmission mechanisms on the relevant stakeholders in Table 3.3. From the user's perspective, the download mechanism results in the least energy consumption E. However, the streaming mechanism provides similar results, especially for larger bit rates b_R. For the video provider, the smallest amount of wasted traffic is generated by using the live mechanism. Again, the streaming mechanism provides a close sec-

Metric / Stakeholder	Download	Live	Provisioning	Streaming
Energy / User	++	−	-	+
Wasted traffic / Video provider	−	++	-	+
Signalling / ISP	++	++	++	Parameter dependant

Table 3.3: Impact of transmission mechanisms on metrics and stakeholder.

ond place. Finally, from the perspective of the ISP, the mechanisms download, live, and provisioning result in the least amount of signalling. The streaming mechanism can be configured in such a way that only relatively small amounts of signalling is generated.

Influence of Buffer Threshold Selection

In this section, we discuss the influence of the lower buffer threshold θ and the buffer size Θ on the energy consumption E, the wasted traffic W and the connection count C for a uniformly distributed user model as discussed in Figure 3.2.2. Considered stop thresholds are in the range of 4 s and 32 s. Lower stop threshold values result in stalling, as the buffer runs empty while the UE is still waiting for the promotion delay to be completed and sufficient amount of data to be downloaded to continue playback. For sake of readability, we show video bit rates b_R for values of $2\,\mathrm{Mbit\,s^{-1}}$, $6\,\mathrm{Mbit\,s^{-1}}$ and $10\,\mathrm{Mbit\,s^{-1}}$ and show the Pareto-frontier, i.e. the set of all parameter combinations where no other parameter combination yields better results for both metrics, of evaluated parameters as a connected line. If only one parameter combination is Pareto-optimal is marked using an arrow.

First, we consider the tradeoff between energy consumption E and wasted traffic in Figure 3.10. We find that, independent of video bit rate b_R, the values found on the Pareto-frontier can be obtained for the smallest considered buffer

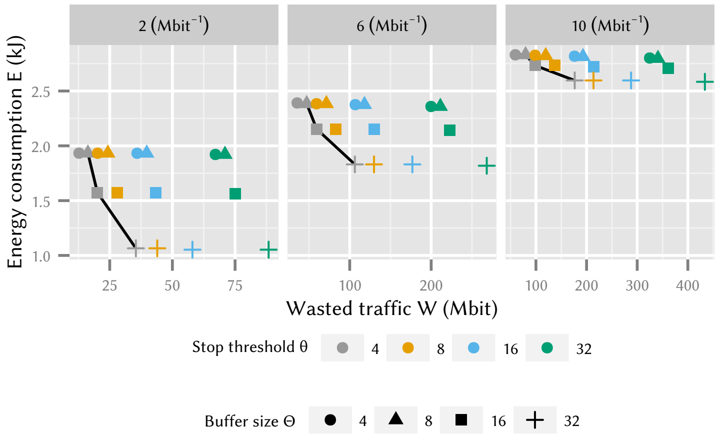


Figure 3.10: Evaluation of the streaming mechanism for varying video bit rates b_R regarding energy consumption E and mean wasted traffic W.

threshold. Increasing the buffer size decreases the energy consumption E at the cost of a higher mean wasted traffic W. Choosing a small lower buffer threshold θ decreases the minimum amount of mean wasted traffic W if the user stops watching a video. Selecting a higher buffer size Θ increases the mean wasted traffic W, as more video can be downloaded and thus wasted if a user stops watching the video. Increasing the buffer size Θ decreases the energy consumption E, as a longer buffer size allows for the video to be downloaded in fewer bursts and each of them is followed by the $T_{\texttt{Idle}}$ timeout where the UE is still in the most energy intensive RRC_`Connected` state. For this tradeoff, we recommend to always use the smallest possible stop threshold generating no stalling. The choice of the buffer size depends on the preference between energy consumption E and mean wasted traffic W, with smaller threshold sizes requiring more energy and higher threshold sizes causing a higher mean wasted traffic W.

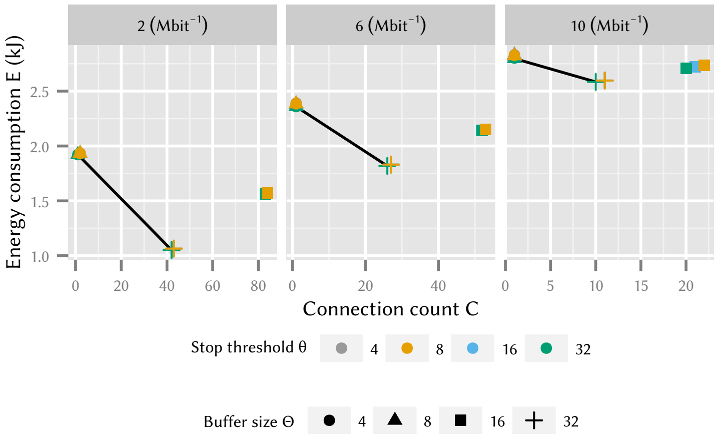


Figure 3.11: Evaluation of the streaming mechanism for varying video bit rates b_R regarding energy consumption E and the connection count C.

In Figure 3.11 we consider both the energy consumption E and the connection count C. For all considered bit rates, the largest possible lower buffer threshold is Pareto-optimal. A tradeoff between energy consumption E and connection count C is possible by varying the buffer size. A small buffer size yields a larger energy consumption E and a smaller connection count C while for a larger buffer size a smaller energy consumption E and a higher connection count C can be achieved.

Finally, in Figure 3.12 we study the tradeoff between the connection count C required per video transmission and the mean wasted traffic W. We observe that for a bit rate b_R of $2\,\mathrm{Mbit\,s^{-1}}$ configurations exist where only one and two connections are required to transmit the complete video. Here, the amount of wasted traffic can be reduced by $82\,\%$ by allowing just one more connection over the whole transmission interval. For video bit rates b_R of $6\,\mathrm{Mbit\,s^{-1}}$ and $10\,\mathrm{Mbit\,s^{-1}}$ only one Pareto-optimal value exists where the video is transmitted

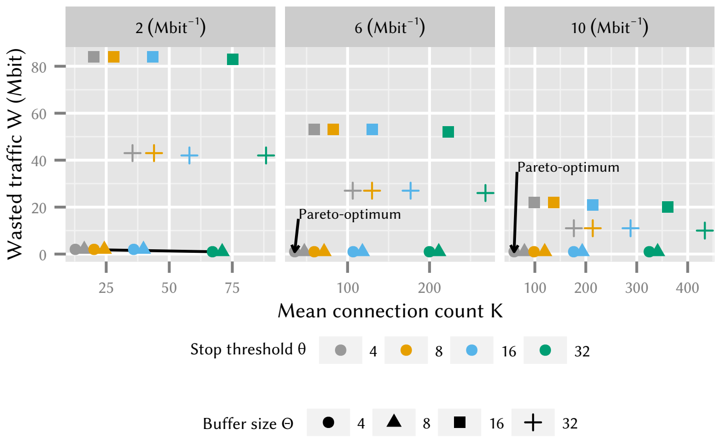


Figure 3.12: Evaluation of the streaming mechanism for varying video bit rates b_R regarding the connection count C and the mean wasted traffic W.

using a single connection.

We find that each of the considered tradeoffs provides Pareto-optimal results and summarise the relevant parameters qualitatively in Table 3.4.

From these results it becomes clear that it is impossible to find Pareto-optimal results for all three tradeoffs. Thus, new decision making policies are required to find optimal results for all participating stakeholders. One such policy is discussed in *Design for Tussle* [48, 77]. Here, the attempts are made to resolve trade-

Tradeoff	Parameter	Optimal value
Wasted traffic vs. Energy	Lower buffer	Low
Connection count vs. Energy	Lower buffer	High
Wasted traffic vs. Connection count	Lower buffer	Low

Table 3.4: Qualitative results of tradeoff analysis.

offs and conflicts at run-time instead of design-time. This way, the requirements of the stakeholders can be adapted to the current situation, e.g. an ISP could require less stringent signalling bounds if the signalling load is currently low in the network. Such functionalities could be implemented in the UEs operating system, which even currently serves as a mediator between the different stakeholders.

3.3 Dimensioning Video Buffer for Specific User Profiles and Behavior

While the previous section implicitly assumed that sufficient bandwidth for video playback is available in order to provide high QoE to the user, this assumption on QoS does not always hold in the real world, for example due to a high user count in the LTE cell, or due to difficult terrain with varying transmission channels.

Traditional QoE management mechanisms [53] consider a mapping function from QoS to QoE obtained from extensive user studies. This MOS homogenises different user ratings due to the use of an average, and do not consider the existence of user groups with distinct preferences. In contrast to the earlier sections in this work, this sections considers tradeoffs between sub-groups of stakeholders, i.e. different user profiles.

To this end, we first introduce models for video playback in Section 3.3.1. Then, we extend available QoE models in order to support parametrisation for user preferences in Section 3.3.2. Finally, we evaluate a set of user scenarios in Section 3.3.3 using both the playback and the QoE model.

3.3.1 Video Playback Model

This section provides a system model for video playback, in order to study the stalling behaviour of HTTP video streaming. We consider the playback of a video consisting of multiple frames, as shown in Figure 3.14. The frames are downloaded in-order and arrive at the client with rate λ while the playback time is given by the video frame rate μ, resulting in an offered load of $a = \frac{\lambda}{\mu}$. Here, a quantifies the available network bandwidth normalised by the video frame rate. We assume a stable system, i.e. that $a < 1$ holds. In order to reduce the number of stalling events during playback, the video player uses a playback buffer. Video playback stops, if less than q frames are currently available for playback and is only resumed if the buffer contains $p = q + d$ frames. The normalised buffer size d^*, in video seconds, relates the buffer size d, which is given in frames, to the video frame rate μ, i.e. $d^* = \frac{d}{\mu}$.

Next, we introduce metrics used to evaluate the influence of the playback buffer parameter selection. The relative amount of time spent in stalling compared to the total duration of the playback process including stalling is given by the stalling ratio R and the number of stalling events normalised by the video length N^*. For the case of finite videos we furthermore consider the stalling duration L which gives the sum of times spent in stalling states during the complete video playback.

$M/M/1$ Queue with pq-Policy

The state of the video playback is characterised by the tuple (i, j), where $i \in \{0, 1\}$ is the playback state of the client, i.e. the video is not played back if i is 0 and the video is played back if i is 1. Furthermore, $j \geq 0$ gives the number of unplayed frames currently available at the client. Furthermore, we give the probability of the playback being in state (i, j) as $x(i, j)$.

We obtain the following equilibrium state equations, based on the state dia-

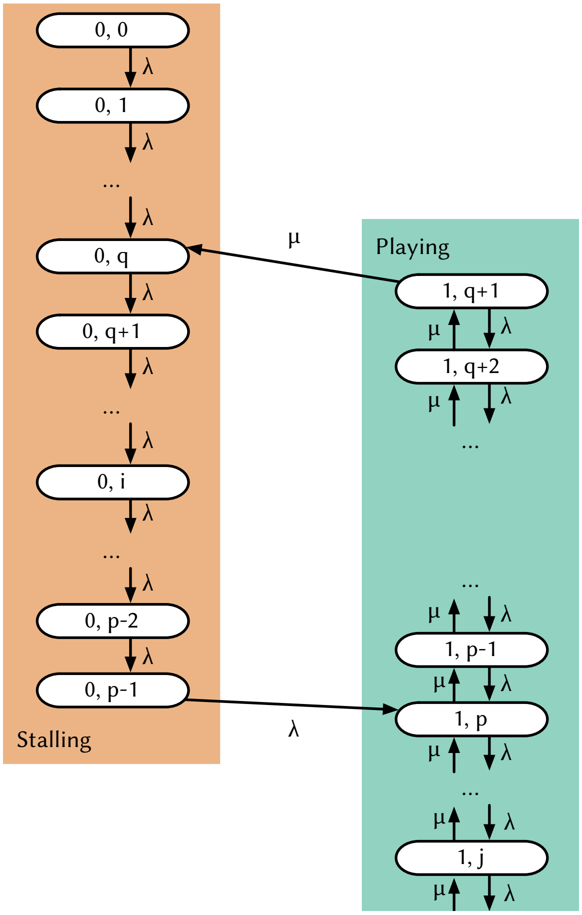


Figure 3.13: System model for the $M/M/1$ queue with pq-policy.

gram given in Figure 3.13.

$$\begin{aligned}
\lambda x(0,0) &= 0 \\
\lambda x(0,i) &= \lambda x(0,i-1) && i \in [1,q) \\
\lambda x(0,q) &= \lambda x(0,q-1) + \mu x(1,q+1) \\
\lambda x(0,i) &= \lambda x(0,i-1) && i \in (q,p) \\
(\lambda+\mu)x(1,q+1) &= \mu x(1,q+2) \\
(\lambda+\mu)x(1,i) &= \lambda x(1,i-1) + \mu x(1,i+1) && i \in (q+1,p) \\
(\lambda+\mu)x(1,p) &= \lambda(x(0,p-1) + x(1,p-1)) \\
&\quad + \mu x(1,p+1) \\
(\lambda+\mu)x(1,i) &= \lambda x(1,i-1) + \mu x(1,i+1) && i \in (p,+\infty)
\end{aligned}$$

State probabilities are obtained using macro state equations and recursive reduction and follow analogously to [65]:

$$\begin{aligned}
x(0,i) &= 0 && i \in [0,q) \\
x(0,i) &= \frac{1-a}{d} && i \in [q,p) \\
x(1,i) &= \frac{a(1-a^{i-q})}{d} && i \in (q,p] \\
x(1,i) &= \frac{a^{j-p+1}(1-a^d)}{d} && i \in (p,+\infty]
\end{aligned}$$

From this, we obtain the stalling ratio R as the probability of being in a stalling state, i.e.

$$R = \sum_{i=0}^{p-1} x(0,i) = 1-a\,. \tag{3.3}$$

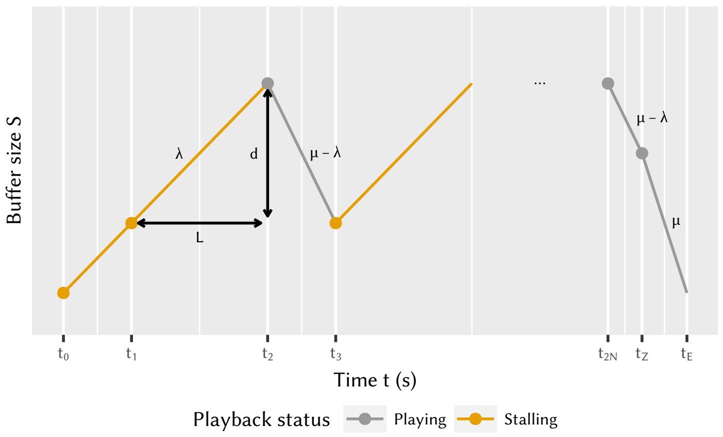


Figure 3.14: Video buffer status evolving over time with constant video bit rate μ and network bandwidth for a finite video of duration T and Z frames.

Mean Value Analysis of Steady State

While the $M/M/1 - pq$ model provides results for infinite length videos, real-world videos however are of finite length. This requires the study of additional metrics, i.e. the number of stalling events during playback N^*. Thus, in this section we derive a mean value analysis of the proposed video playback model according to Figure 3.14.

Assuming that the initial download begins at t_0 and new frames arrive with rate λ at the client. The number of frames in the buffer S exceeds q the first time at t_1. At time t_2, the threshold of p is reached for the first time and playback begins. While the download of new frames continues with rate λ, frames are played out with rate μ, resulting in a buffer change with rate $\mu - \lambda$. Thus, the number of buffered frames S reaches q at time t_3. This process repeats, resulting in an alternating chain of stalling and playback phases.

In this analysis, we consider the steady state, i.e. especially neglecting the time $t_1 - t_0$. First, we consider the time required for the buffer to fill from q frames to p frames, i.e. obtaining d frames while no playback is occurring. This time depicts the average duration L of a single stalling event.

In Figure 3.14 this is given as the time between t_1 and t_2, and we get

$$L = t_2 - t_1 = \frac{p-q}{\lambda} = \frac{d}{\lambda} = \frac{d^*}{a} \, .$$

The average stalling length L only depends on the actual buffer size d and the network bandwidth λ.

Next, we consider the time required for the number of frames S in the buffer to decrease from p to q, i.e. the time between t_2 and t_3,

$$t_3 - t_2 = \frac{d}{\mu - \lambda},$$

which represents the time of uninterrupted playback between each stalling event.

Combining these two equations we get the time between two stalling event as

$$t_3 - t_1 = (t_3 - t_2) + (t_2 - t_1) = \frac{\mu d}{(\mu - \lambda)\lambda}.$$

The stalling ratio R follows as

$$R = \frac{t_2 - t_1}{t_3 - t_1} = 1 - a,$$

yielding the same result as in Equation 3.3.

Finally, we can obtain the number of of stalling events normalised by video duration by analysing the busy periods of the system. Here, the mean idle period is given by $L = \frac{d}{\lambda}$.

For the mean busy period B we obtain

$$\frac{B}{B+L} = 1 - R = a\,,$$

which yields

$$B = \frac{a}{1-a}\frac{\lambda}{d}$$

and in turn can be used to obtain the normalised number of stalling events

$$N^* = \frac{1}{B} = \frac{\mu - \lambda}{d} = \frac{1-a}{d^*}\,.$$

This equation can also be derived by considering $N^* = \frac{1}{t_3 - t_2}$. While N^* relates the number of stalling events to the video duration, the stalling frequency F denotes the number of stalling events per unit time. It holds $F = \frac{1}{t_3 - t_1} = aN^*$ which can be obtained as $F = x(0, p-1)\lambda$ to weight the state probability of player state change $x(0, p-1)$ with the network arrival rate λ. From an end user's point of view, the metric N^* is of higher importance.

Beside the network bandwidth λ and the video bit rate μ, the metric N^* of stalling events depends only on the video buffer size $d = p - q$, but not on the concrete values of p and q in the steady state.

Mean Value Analysis of Finite Videos and User Aborts

As we will see later in Section 3.3.3, the steady state analysis is sufficient to dimension the buffer. However, in practice, playback is finite, as either the video is of finite length T, or due to the fact that a user aborts playback after a number of T seconds. This behaviour is shown in Figure 3.14.

We do not consider the time until the initial playback, i.e. the time between t_0 and t_2 as stalling, since it has a much lower impact on the perceived quality than stalling [78] and it only depends on the network bandwidth λ and thus is not subject to optimisation. First, we consider the case where the user plays back the complete video. Given the network bandwidth λ and a video of Z Frames, the required download time for the complete video is $t_Z = \frac{Z}{\lambda}$. Within t_Z there are N phases of stalling and playback and each phase is of duration $t_3 - t_1$, i.e.

$$N = \left\lfloor \frac{t_Z - t_1 + t_0}{t_3 - t_1} \right\rfloor .$$

Next, we consider the case where the user aborts playback of the video after T seconds of video have been watched. Here, the number of stalling phases is given as

$$N = \lfloor T/(t_3 - t_2) \rfloor ,$$

rounding down as we do not consider the initial delay before playback as stalling. Again, we can obtain the number of stalling events normalized by video length as $N^* = \frac{N}{T}$.

3.3.2 YouTube QoE Model

This section introduces QoE models for YouTube video playback. First, we extend the QoE mapping function introduced in [53] in order to support user preferences regarding sensitivity to stalling duration and number of stalling events. Then, we provide a parametrised mapping function allowing for user preferences regarding initial delay. Finally, we combine the proposed mapping functions.

Stalling QoE Model

The QoE of HTTP streaming depends mainly on the actual number of stalling events N for a video of duration T and the average length L of a single stalling event. A QoE model combining both key influence factors into a single equation $f(L, N)$ is provided in [53] and found to follow the IQX hypothesis [79] describing an exponential relationship between the influence factors and QoE. In particular, the model function returns MOS on a 5-point absolute category rating scale with 1 indicating the lowest QoE and 5 the highest QoE:

$$f(L,N) = 3.5e^{-(0.15L+0.19)N} + 1.50. \tag{3.4}$$

Due to well known rating scale effects, the model in Equation 3.4 has a lower bound of 1.50, as users avoid the extremities of the scale called *saturation effect*, see e.g. [80]. In contrast, if the video is not stalling, no degradation is observed and users rate the impact of stalling as 'imperceptible', i.e. a value of 5.

It has to be noted that the model function in Equation 3.4 is based on subjective user studies with videos of duration up to $T = 30\,\text{s}$. For other video durations, the normalised number $N^* = \frac{N}{T}$ of stalling events has to be considered which requires to adapt the parameters $\alpha = 0.15$ and $\beta = 0.19$ in Equation 3.4, respectively.

As the goal of our investigation is the analysis of the impact of different user profiles, we parametrise the function in Equation 3.4 with parameters α and β and conduct a parameter study on their impact. For the sake of simplicity, we normalise the QoE value to be in the range $[0;1]$ and use the normalised number of stalling events N^*. As a result, we arrive at Equation 3.5 below as a parametrised QoE model Q_1 to quantify the impact of stalling on QoE for different user profiles expressed by α and β. Thereby, the parameter α adjusts the sensitivity of the user to the stalling duration $L \cdot N^*$, while β quantifies the sensitivity of the user to the actual number of stalling events, i.e. the video interruptions. Therefore, we will also use the term *duration parameter* and *interruption parameter* for α and β, respectively.

The model function Q_1 in Equation 3.5 has the same form as Equation 3.4 and follows the IQX hypothesis, but allows to investigate different user profiles:

$$Q_1(L,N^*) = e^{-(\alpha L+\beta)N^*}. \tag{3.5}$$

For example, some users may suffer stronger from interruptions which is then adjusted by a higher value of β. Thus, a user profile can be expressed in terms of different values of the duration parameter α and the interruption parameter β

Initial Delay QoE Model

Another impairment on HTTP streaming QoE are initial delays before the video play-out can start for the first time. The impact of initial delays T_0 is modelled by the function given in Equation 3.6, model parameters are obtained from subjective tests [57], yielding

$$g(T_0) = -0.963 \cdot \log 10(T_0 + 5.381) + 5. \tag{3.6}$$

The results in [57] show that the impact of the initial delay is independent of the video duration which was either 30 s or 60 s in the user tests. Further, it was observed that users have a clear preference of initial delays instead of stalling and that service interruptions have to be avoided in any case, even at costs of increased initial delays for filling up the video buffers.

For the sake of simplicity, we normalise the function in Equation 3.6 to obtain the QoE model Q_2 for initial delays T_0, so that Q_2 returns values in $[0;1]$ and that $Q_2(0) = 1$ holds, by adding the term $\gamma \log 10\,(c)$.

The user profile is parametrised with the parameter γ determining the impact of initial delays on the user QoE. The constant $c = 5.381$ is taken from Equation 3.6 defining the shape of the curve. Since the logarithm is not bounded, only positive values are considered to ensure $Q_2(T_0) \in [0;1]$:

$$Q_2(T_0) = -\gamma \log 10\,(T_0 + c) + \gamma \log 10\,(c) + 1.$$

Combined QoE Model

For dimensioning the video buffers, we are interested in a QoE model which considers both the impairments due to stalling and due to initial delays of the video play-out. However, to the best of our knowledge no combined model exists so far which has been validated by proper subjective user studies. Therefore, we suggest the following model Q. Since the impact of stalling events clearly dominates the user perception [57, 72], we consider the following rationale for the combined QoE model. A user facing an initial delay T_0 experiences a QoE value of $Q_2(T_0)$. If additional stalling events occur, this will lower the QoE further. Thus, $Q_2(T_0)$ is the upper bound of QoE. For N^* stalling events with an average length L, the QoE will be further decreased by $Q_1(L, N^*)$.

An additive QoE model for non-adaptive HTTP streaming which is referred to as buffer-related perceptual indicator is recommended in [81]. This model follows the same rationale above, start from the maximum QoE value which is $1 = Q(0, 0, 0)$, subtract the degradation from stalling $1 - Q_1(L, N^*)$ and $1 - Q_2(T_0)$ stemming from initial delay.

Then, we arrive at the following additive QoE model Q used in the following analysis:

$$\begin{aligned} Q(T_0, L, N^*) &= 1 - (1 - Q_1(L, N^*)) - (1 - Q_2(T_0)) \\ &= Q_1(L, N^*) + Q_2(T_0) - 1. \end{aligned} \tag{3.7}$$

3.3.3 QoE Study for Typical User Scenarios

Based on the playback model introduced in Section 3.3.1 and the parametrised QoE user model proposed in Section 3.3.2, this section studies three typical user scenarios. We discuss optimal choices for buffer size depending on user preferences and highlight the impact of buffer choices neglecting the user preference.

Watch Later Scenario

In the *Watch Later* scenario, a user requests a video, but the user does not expect that the video play-out starts immediately. This may be the case, for example, when the user wants to watch an HD at a later time and expects low network bandwidth later. During that initial delay, the user may opt do something else, e.g. opening another web page in a parallel tab in the browser. Thus, QoE is not affected by initial delays and we only need to consider Q_1 in Equation 3.5.

In the steady state, we have $L = \frac{d}{\lambda}$ and $N^* = \frac{\mu-\lambda}{d}$ and we obtain the following QoE relation:

$$Q_1(L, N^*) = e^{(\mu-\lambda)(\frac{\alpha}{\lambda}+\frac{\beta}{d^*})} = e^{-\alpha\frac{1-a}{a}-\beta\frac{1-a}{d^*}}. \tag{3.8}$$

Since the QoE function in Equation 3.8 is strictly monotonically increasing in the normalised buffer size d^*, the optimum is achieved for

$$Q_+ = \lim_{d^*\to\infty} Q_1(L, N^*) = e^{-\alpha\frac{1-a}{a}}.$$

Thus, the QoE value only depends on the parameter α, i.e. the total stalling duration, in the limit. To see for which buffer size we are close to the optimum, we consider the relative difference $\Omega = \frac{Q_+ - Q_1(L,N^*)}{Q_+}$ when it is less than $\Omega = 5\,\%$. This is true for any $d^* > -\beta\frac{1-a}{\log(1-\Omega)}$.

For $\beta \in \{0.05, 0.2\}$, a small buffer size of $d^* > 4\,\mathrm{s}$ is already sufficient to be close to the optimum Q_+ for any offered network condition a. For users extremely sensitive to stalling, e.g. for $\beta = 0.8$, buffer sizes up to $15\,\mathrm{s}$ are required. However, a buffer of $4\,\mathrm{s}$ is sufficient for a relative difference to the optimum of $20\,\%$. In general, the larger the buffer size the better the obtained QoE is in this scenario. In practice, a buffer size of $4\,\mathrm{s}$ is a good choice.

Default Video Streaming Scenario

In the case of normal streaming, the user wants to watch a video immediately for a long period of time. In contrast to the *Watch Later* scenario, the initial delay impacts the QoE in the *Watch Now* scenario according to Equation 3.7.

Figure 3.15 shows QoE depending on the buffer size d^* for the *Watch Now* scenario and different user profiles in a network situation $a = 0.5$ leading to a stalling ratio $R = 0.5$.

QoE optima exist for finite buffer size, if the impact of the initial delay is taken into consideration. We notice that for users more sensible to stalling duration,

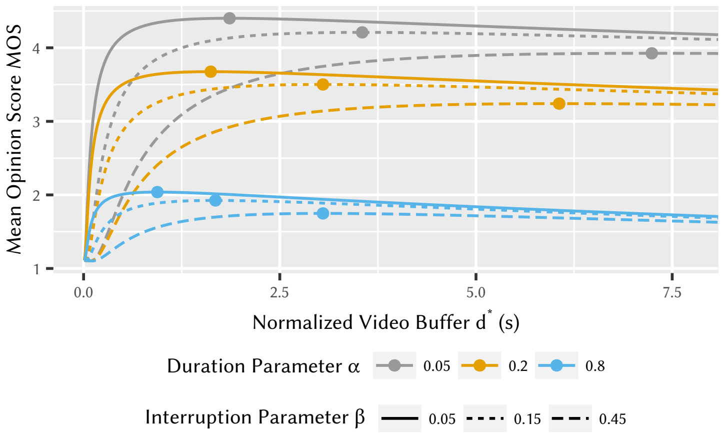


Figure 3.15: Dimensioning of buffer size in the Streaming Scenario for available network bandwidth of $a = 0.5$. Maxima marked as dots mainly depend on β.

i.e. higher values of α, experience higher QoE, but this has no significant impact on the optimal buffer size. In contrast, for users sensitive to the number of stalling events, i.e. for higher values of β, we observe different optima for the buffer size. Therefore, we can neglect the interruption parameter α when optimising the buffer size with regard to the QoE. A buffer size less than $0.5\,\mathrm{s}$ results in a severe loss of QoE for all users. A buffer size of $2\,\mathrm{s}$ to $4\,\mathrm{s}$ offers a good QoE for the average user and any sensitive user. Increasing the buffer size further decreases the QoE slightly.

Video Browsing Scenario

In the case of the *Video Browsing* scenario, the user watches a video for a short period of time. This includes cases such as, viewing a short video completely, viewing a short part of a long video or skipping ahead in a video frequently, thus watching multiple short parts of a video. In this scenario, a steady state can not be assumed due to the short watching duration. Since we know from the previous section that α and β have only a marginal impact on the optimal QoE, we consider only the default parameters $\alpha = 0.15$ and $\beta = 0.2$ in the following. However, for Video Browsing, the impact of the initial delay may be more important for the user. Therefore, we consider two different types of delay sensitive users with $\gamma = 0.2$ as well as a more delay sensitive user with $\gamma = 0.8$.

In Figure 3.16, the impact of the buffer size on the QoE is depicted for the case that the video is aborted after the first $10\,\mathrm{s}$ using a logarithmic x-axis. We consider two different network scenarios with an offered load of $a = 0.8$ and $a = 0.95$. Multiple local QoE maxima exist independently of γ, which appear when the number of stalling events change. For different values of β these maxima occur at the same buffer size. Therefore, we can ignore β in this scenario. The local minima exist at the buffer size for which the last stalling event has the smallest possible length. The results for the steady state are also included and we observe that the steady state provides a lower bound for the finite buffer results.

Thus, the steady state can be used to perform worst case buffer dimensioning.

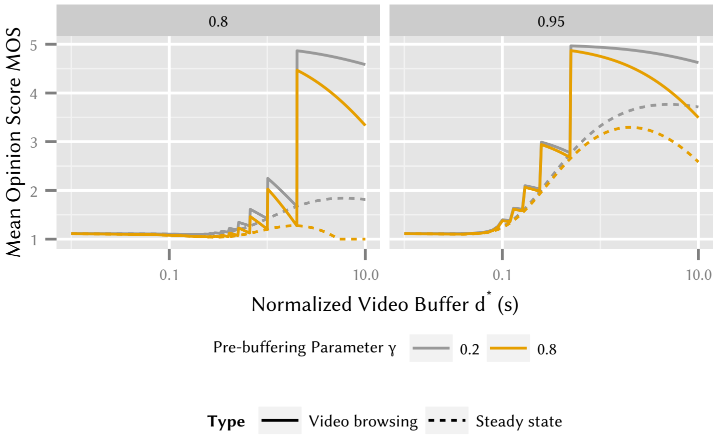


Figure 3.16: Dimensioning of buffers for Video Browsing users with varying QoE sensitivity to initial delays. Users abort the video after 10 s.

For very low offered loads a, e.g. $a = 0.1$ which is not shown due to scale, the QoE is very low for both the steady state and the finite case. Thus, Video Streaming and especially *Video Browsing* is not desirable in this case. However, for larger buffer sizes, the difference between the local maxima and the steady state increases. Nevertheless, in those cases, the initial delay exceeds tens of seconds. So this scenario can not be described as realistic *Video Browsing*.

In general, if the exact viewing length of a video was known, e.g. short videos will be watched completely, the buffer size could be set so that the QoE lies at a local maximum which is independent of γ. However, this method can result in a severe loss of QoE, depending on γ if the user aborts earlier, as the actual QoE loss significantly depends on γ. In practice, a buffer size of 1 s and 2 s is recommended for Video Browsing. If the buffer size is set too large, γ determines again the actual QoE loss. For larger buffer sizes, the sensitivity γ to initial delays strongly influence the QoE.

3.4 Cloud File Synchronisation Services

While the previous sections studied video playback, this is far from the only application deployed in today's networks. Another prominent type are file synchronisation services. They are offered by a multitude of providers, e.g. Google Drive, and Microsoft OneDrive, and the current leader in the field Dropbox. A general study of the QoE influence factors of file storage services is undertaken in [82]. Here, the authors provide a model allowing for the evaluation of cloud service providers according to a variety of metrics, including bandwidth, latency, and response time. Due to its popularity, this section focuses on Dropbox.

In [83] the authors study the main impact factors on QoE of Dropbox users. They find that the main impact factor for user QoE is the waiting time for file synchronisation. Thus, this is the main factor the cloud storage operator will dimension the service for, in order to ensure a high user satisfaction. However, the operator of the file synchronisation service is not the only stakeholder in this scenario. As in earlier sections, the signalling traffic imposed by application traffic is of concern to network providers. Furthermore, the services users are, as discussed in Chapter 2, interested in reducing the power drain due to network connectivity in order to increase their devices battery life.

In Section 3.4.1 we first propose a model in order to evaluate the Dropbox service, define relevant metrics for each stakeholder as well as different scheduling algorithms used to trigger file synchronisation. Then, in Section 3.4.2 we perform network measurements using the PlanetLab [25] research network to be used as input for our model. Finally, in Section 3.4.3 we evaluate our model using a non-stationary discrete event simulation for all considered metrics and scheduling algorithms.

3.4.1 System Model for File Synchronisation using Dropbox

This section first provides a general overview over the Dropbox service architecture and introduces the considered use case. Then, we propose the cloud storage

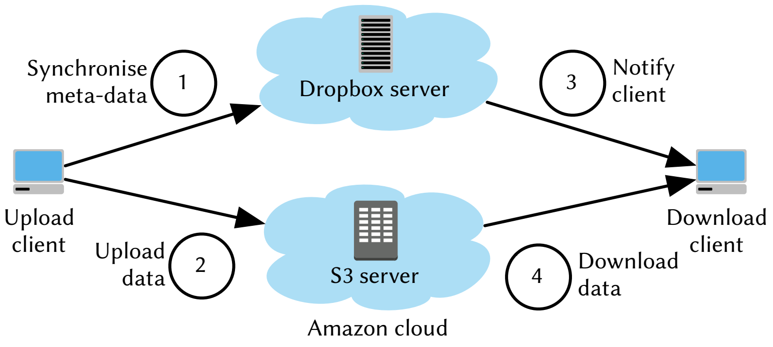


Figure 3.17: Dropbox file storage and retrieval process.

model and metrics used in this analysis. Finally, we discuss a set of scheduling mechanisms used to start the file synchronisation process.

The authors of [84] provide a first study of the *Dropbox* architecture, which is schematically depicted in Figure 3.17 and used as a basis for the model under study in the remainder of this section.

The *Dropbox* infrastructure consists of two main components: *a*) a storage cloud based on Amazon's Elastic Compute Cloud and Simple Storage Service, and *b*) control servers directly maintained by *Dropbox* Inc. The control servers store meta information about the current state of the files in the *Dropbox* folders and trigger synchronisation processes on the clients.

A file synchronisation can basically be described in five steps. As soon as the new file is added to the *Dropbox* folder of the uploading client, a preprocessing step is triggered and the meta information for the file are generated, respectively updated. This information is then synchronised with the control servers (1) and the file itself is uploaded to the storage cloud (2). After the file has completely been transferred to the storage cloud, all connected clients are notified about the update (3) and start downloading the new file (4).

Use Case: Photo Uploading

In this section we consider the synchronisation of images from a digital camera to a mobile UE via a cloud storage provider. Real world examples of this scenario are, e.g., taking photos of a live event and transferring them to a picture agency, or shooting private holiday images.

The user took a finite set of pictures with a wearable device like Google Glass or a smart camera, e.g. a Nikon Coolpix S800c or SAMSUNG CL80. The camera is then connected via a Personal Area Network (PAN) with a mobile UE, for example a laptop with a data card or a smartphone, to store the images on the mobile device. The UE uses broadband wireless Internet access technology and runs software provided by the cloud storage provider in order to synchronise the images with the cloud storage. Finally, the scenario includes a remote client, which is connected using a wire line connection and downloads the images from the cloud.

For the evaluation we consider a specific realisation of the use case described above. For the role of the cloud storage provider we consider *Dropbox*, Bluetooth is used as the technology for establishing the PAN, and LTE is used as the wireless broadband access technology.

In the considered scenario the interests of two stakeholders are impacted. The first stakeholder, the end user, has two contradicting requirements on the system. On the one hand, the images should be synchronised as fast as possible. This requires a fast and permanent Internet connection of the UE, which in turn is very power intensive. On the other hand, the power drain of the mobile device should be minimised to enable a long battery life time. The second stakeholder, the mobile network provider, wants to minimise the signalisation overhead in the network [23, 27] caused by short time connections. Here, an optimisation problem arises to find a practical solution for all three requirements. In order to analyse this problem, we use a simulation model of the file synchronisation process, which is described in the following.

Cloud Storage Model and Performance Metrics

The proposed simulation model is schematically depicted in Figure 3.18 and based on the findings of [84] described in Section 3.1.

We assume that the user has taken pictures of varying file size distributed with S_I. These pictures are transferred from the camera to the mobile device using the PAN with a constant bandwidth B_P. Due to the limited bandwidth B_P of the PAN device, the inter-arrival times of images at the *Dropbox* shared folder of the mobile device can be calculated by $t_I = \frac{S_I}{B_P}$.

As soon as the image is fully copied to the *Dropbox* folder, the generation of the meta data introduces a preprocessing delay, which we refer to as client preparation time C. To evaluate different strategies optimising the overall waiting time, power drain, and signalisation traffic we include a scheduling component. This component implements different algorithms, which trigger sending the images currently held available in the scheduling component to the component responsible for transmission.

Next, we consider the LTE UE used for image upload. Due to the specification of the LTE standard [28], the upload component can, at any point in time, either be connected to the mobile network or disconnected. If the UE is currently disconnected, and a new image for upload arrives, the connection process is triggered and completed after a startup delay $\sigma = 0.26\,\text{s}$. Once the UE is connected, arriving images are transmitted in order. The transmission, i.e. service time, of an image depends on the size of the image currently being uploaded as well as the upload bandwidth B_U. As only one image is transferred at once, waiting images are stored in a queue of infinite size. If the UE is idle for more than $\tau = 11.576\,\text{s}$, the device disconnects from the network.

After the image has been successfully uploaded to the storage servers, a server side preprocessing phase starts, before the file transfer to the downloading client starts. This server side preprocessing again introduces an additional delay, the server preparation time S, in the synchronisation process. Finally, the image is downloaded by the wire line client. Again, the duration is calculated based on the size of the image and the available download bandwidth B_D.

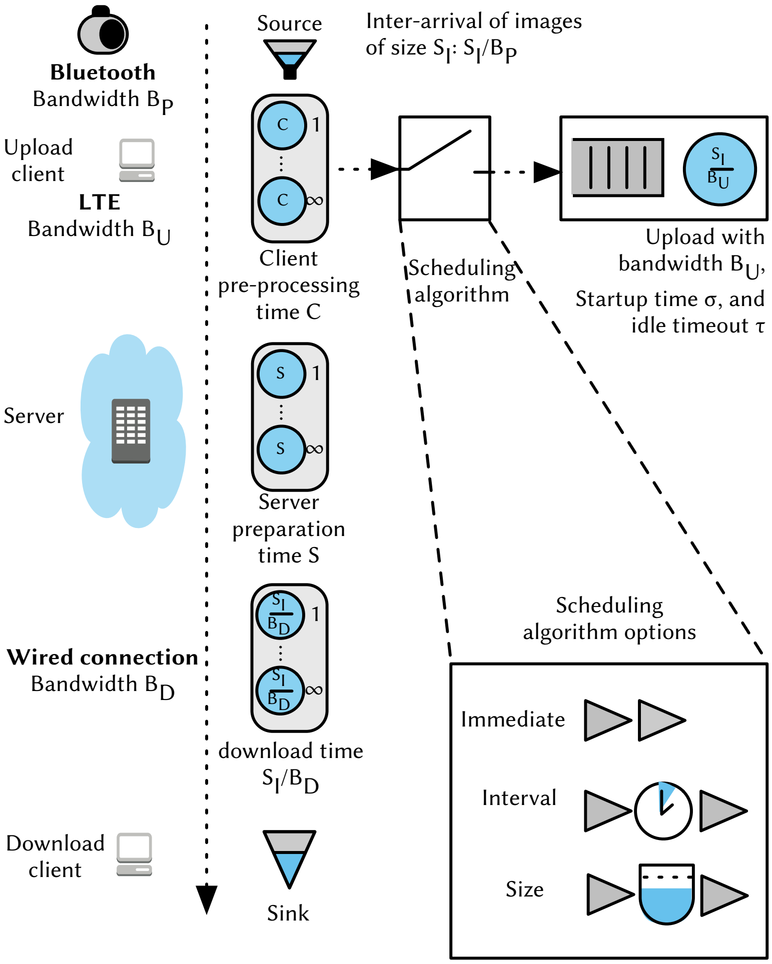


Figure 3.18: Considered synchronisation process model.

Next, we discuss the metrics used to evaluate the performance of the scheduling algorithms under consideration. First, we consider the mean synchronisation time Σ, i.e., the time between the generation of images and the completion of the download. This metric accounts for the desire of end users to synchronise images in a short amount of time. Secondly, we study the relative amount of time the UE is disconnected Δ. As the UE consumes more power in the connected state, the user is generally interested in scheduling mechanisms which ensure that the device is only connected if required [37]. This measure also enables a more general evaluation than the actually consumed power, as the concrete power drain differs significantly for each device. Finally, we evaluate the number of transitions K between the connected and disconnected states. As discussed in Section 2, frequent state transitions stress the network due to increased signalling. Thus, scheduling algorithms with a small number of transitions would be favoured by network operators.

Considered Scheduling Algorithms

We use different scheduling strategies in our model to control the uploading of the files from the mobile client. These mechanisms in turn affect the synchronisation time, the power drain, and the generated signalling traffic.

The most basic strategy of handing the upload is to immediately send new files, as soon as the meta data is generated. We refer to this as the *Immediate* strategy and will use this as base line for all comparisons in the evaluations. The other two strategies considered are based on a temporal, respectively a size threshold. Using the *Interval* scheduling, the client checks periodically, according to an interval T_i, if new files have been marked for synchronisation. If new files are present, they are synchronised to *Dropbox*. Files which could not be sent within the current interval will automatically be added to the file batch for the next interval. The last scheduling mechanisms uses a threshold T_s based on the overall *Size* of the images not yet synchronised. If the threshold is crossed, an upload is triggered.

3.4.2 Measurement of Dropbox Performance Metrics

The model presented in Section 3.4.1 requires several input parameters which are obtained from measurements and described in this section. First, we give an overview over the measurement setup implemented in the distributed testbed PlanetLab [25]. Then, we present measurement results as well as fitted distributions for both up and download bandwidth as well as for the time used by the cloud service to prepare data prior to the download. Finally, we derive an image file size distribution by analysing a large set of digital photos.

Bandwidth and Preparation Times

We obtain a PlanetLab slice containing all momentarily available nodes in February 2014 and discard every node not responding to `ping` or `ssh` within a 20 s interval. On the remaining 87 nodes we install our measurement setup. This includes two instances of the *Dropbox* client on each host, linking them to a specially created *Dropbox* account and two different directories. Furthermore, we disable the *LAN-Synchronisation* feature for both clients. After ensuring that both shared directories are empty, we create a file with randomly generated content of 10 Mbit size, unique per node. Files are unique in order to compensate for caching algorithms by *Dropbox*, as the client calculates a checksum of the file prior to uploading and only uploads the file if no duplicate file is already stored in the account, in order to conserve bandwidth. Further, the randomly generated content ensures that no significant compression results can be achieved before uploading, resulting in comparable results for the time required to upload the files. After the file is created, we start `tcpdump` and *symlink* the file to the first directory shared via *Dropbox* while taking note of the *initial timestamp* of the symlink. As the complete file has finished downloading and appears in the second *Dropbox* directory, we note the current *final timestamp* and stop `tcpdump`. Finally, we retrieve the traffic dump and the recorded timestamps and reset the measurement setup.

This process is repeated 8 times for all available PlanetLab nodes. Based on the

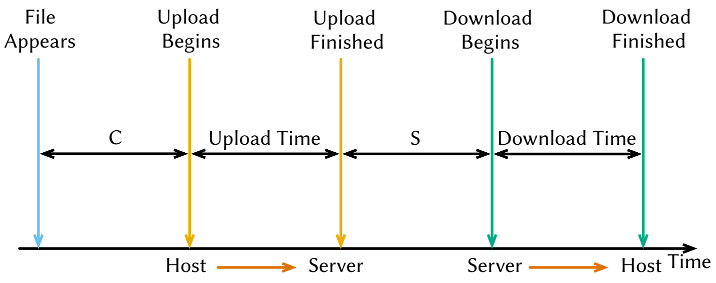


Figure 3.19: Setup used to peform measurement.

two recorded timestamps as well as the traffic dump, we calculate the required values for the model as shown in Figure 3.19. The time between the *initial timestamp* and the first packet sent to the *Amazon S3* storage server is considered the client preparation time C.

The upload time t_u is given by the time between the first and the last packet uploaded to the storage server and is used to calculate the mean upload bandwidth $B_U = 10\,\mathrm{Mbit}/t_u$. The server preparation time S is given by the time between the last packet uploaded and the first packet downloaded from the storage server. Finally, we calculate the mean download bandwidth B_D similar to the upload bandwidth by considering the time between the first downstream packet from the storage server and the completion of the file download.

In Figure 3.20 we show the mean upload and download bandwidth obtained from our measurements, as well as fitted distributions. For the fit we considered a set of different possible distributions, including exponential, log-normal, gamma, Pareto, and Weibull. We found that none of the considered distributions provided an adequate fit due to the slope at the $7\,\%$ quantile for the upload, respectively $25\,\%$ quantile for the download bandwidth. To adapt for this behaviour, we consider a 2-*hyper log-normal* distribution [85] with partitions at 0.07 or 0.25. After splitting, the random variables are fitted to log-normal

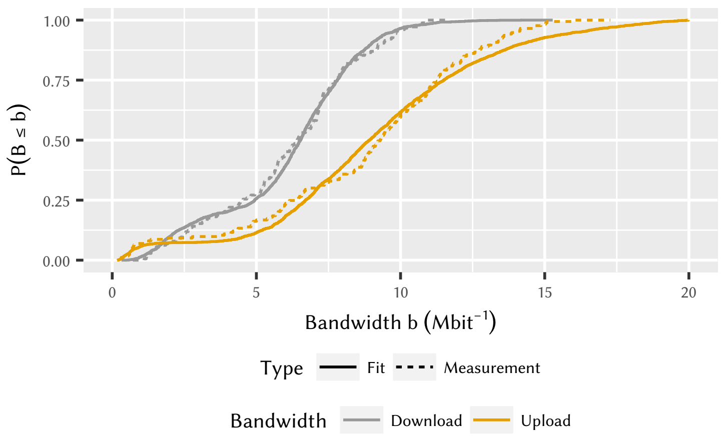


Figure 3.20: Measurement and corresponding fit for upload bandwidth B_U and download bandwidth B_D.

random variables using `fitdistrplus`[2] for the *R* language. The resulting parameters for the upload bandwidth B_U and download bandwidth B_D can be found in Table 3.5, where μ_1 and σ_1 are the location and scale parameters for the *lower* part of the compound distribution and μ_2 and σ_2 are the location and scale of the *upper* part of the compound distribution.

Next, we consider the client and server preparation times. As our intent is to evaluate the performance of different scheduling mechanisms for the upload phase, we require only the amount of time C used for computing hashes of the files considered for upload without considering the time used by the scheduling mechanism. To obtain an approximation, we use the *minimum* of all observed upload preparation phases $C = 1.32\,\text{s}$. For the server preparation time S we obtain a fit, finding that a Log-normal distribution provides the best result of

[2] `https://cran.r-project.org/web/packages/fitdistrplus`, Accessed: November, 21st 2015

Random variable	Split	μ_1	σ_1	μ_2	σ_2
B_U	0.07	13.44	0.49	16.10	0.37
B_D	0.23	14.63	0.51	15.81	0.21
S	–	1.35	0.41	–	–
S_I	0.35	14.17	0.54	15.24	0.33

Table 3.5: Distribution parameters for considered random variables.

.

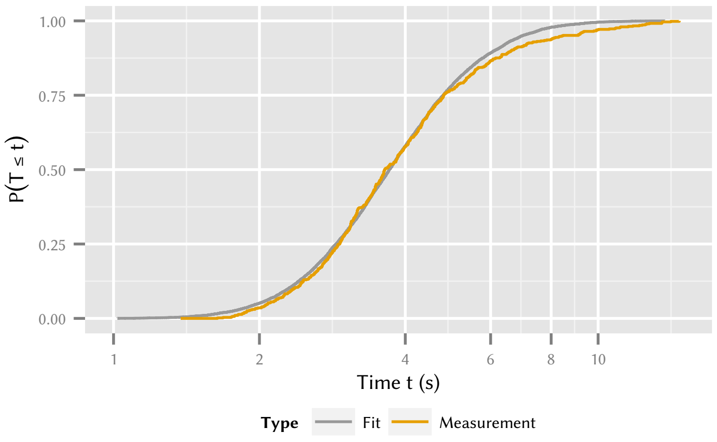


Figure 3.21: Measurement and fit of server preparation time S.

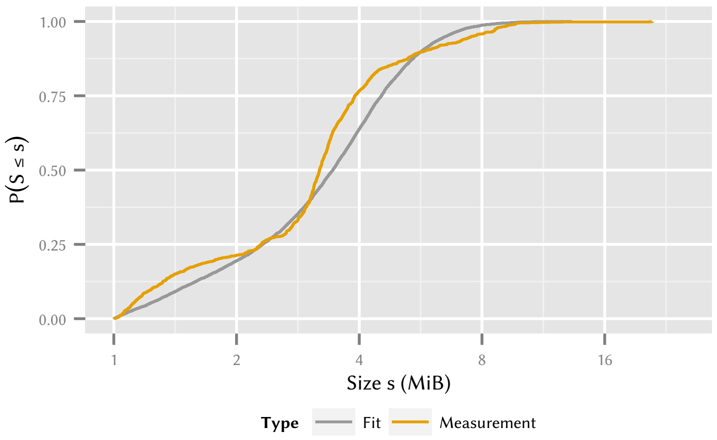


Figure 3.22: Image file size distribution S_I and corresponding fit.

the considered distributions, as shown in Figure 3.21. The parameters of the resulting distribution are given in Table 3.5.

Image File Sizes

In order to obtain a representative random variable for the size of image files, we evaluate a set of 1375 pictures of varying image quality taken by different cameras. We record the file-size and evaluate the quality of fits using different random variables as shown in Figure 3.22. We find that similar to the upload and download bandwidth, a 2-Hyper Log-normal distribution provides best results and present the distribution parameters in Table 3.5.

3.4.3 Evaluation of Considered Scheduling Algorithms

In order to evaluate the proposed model we use the OMNeT++[3] simulation framework. To analyse the impact of the different algorithms we study the metrics introduced in Section 3.4.1. We evaluate the waiting time Σ until a file is retrieved at the downloading client, the relative time the mobile client stays disconnected Δ during the synchronisation process, and the number of connection K during the synchronisation process to estimate the signalling overhead. For the *Interval* scheduling algorithm, we vary the interval length T_i from $1\,\mathrm{s}$ to $512\,\mathrm{s}$ in powers of two. The threshold T_s for the *Size* algorithm is analysed for values from $1\,\mathrm{MB}$ to $512\,\mathrm{MB}$ in the same manner. The *Immediate* algorithm is not parametrised.

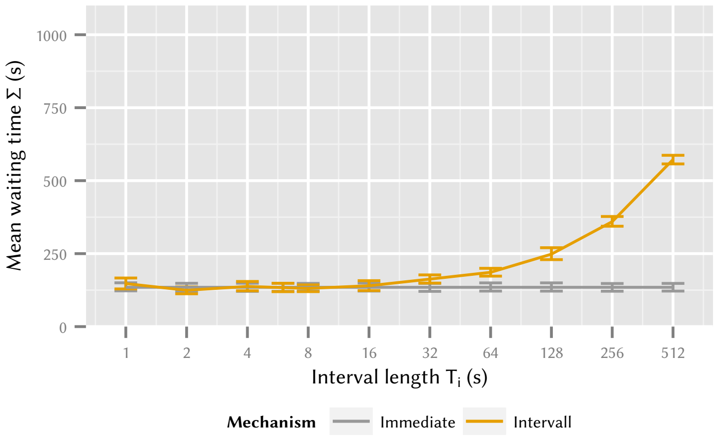
In the simulated scenario, we assume a user synchronising $n = 1000$ files from the camera to the downloading client. For each parameter set we perform 100 repetitions.

Waiting Time

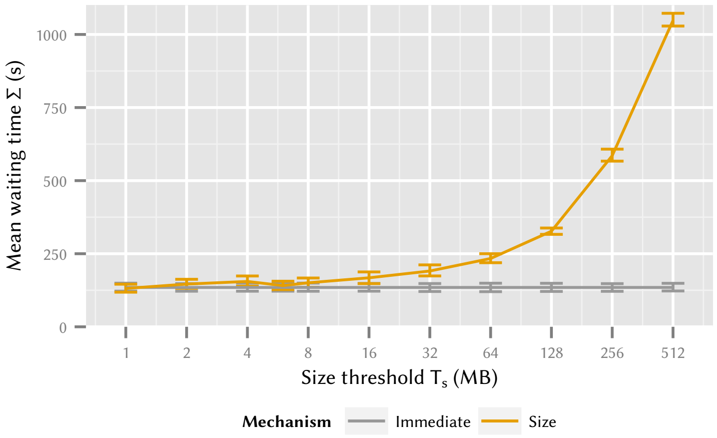
First, we analyse the mean waiting time Σ required to transfer a picture from the camera to the wire-lined download client for the different scheduling algorithms and different parameter sets. The mean waiting times Σ and the corresponding $95\,\%$ confidence intervals are shown in Figure 3.23. For most of the derived mean values, the confidence intervals are not visible due to their small size.

Figure 3.23a shows the results for the *Interval* algorithm, Figure 3.23b shows the results for the *Size* mechanisms. In Figure 3.23a the x-axis shows the length of the interval T_i in s between sending newly added files, in Figure 3.23b the axis shows the required cumulated size in MB of new files before an upload is triggered. The y-axis in both figures shows the mean waiting time Σ in s. The result of the *Immediate* algorithm is added in both figures as a reference. Note that, the mean waiting time Σ for this algorithms is independent of the parameters used for the other two algorithms, as files are always uploaded immediately

[3] `http://www.omnetpp.org/`, Accessed: November, 21st 2015



(a) *Algorithm Interval, different interval lengths T_i*



(b) *Algorithm Size, different threshold sizes T_s*

Figure 3.23: Comparison of algorithms with regard to waiting time Σ.

after they have been copied to the *Dropbox* folder.

Figure 3.23a shows that the mean waiting time Σ depends on the interval length T_i of the *Interval* algorithm. For interval lengths T_i smaller than 8 s, the waiting times do not differ significantly from the mean waiting times Σ obtained by the *Immediate* algorithm. This can be explained by the average file size of $3.5\,\mathrm{MB}$ of the images and the assumed average Bluetooth transmission rate of $0.5\,\mathrm{Mbit\,s^{-1}}$, which results in an average inter arrival time of 7 s. For interval lengths T_i less than 7 s, there is almost always an image in the upload queue so that the algorithm performs similar to the *Immediate* strategy. For interval lengths T_i larger than 7 s, the average waiting time increases for the *Interval* algorithm. Here, the files are already preprocessed and accumulate in the uploading queue until the next batch upload starts resulting in an increased mean waiting time Σ. For very large values of the interval lengths T_i, the mean waiting time Σ is dominated by the interval length T_i. This means that the mobile client's mean waiting time Σ before starting the upload is much longer that the upload duration of the images. Thus, the mean waiting time Σ converges to the interval length T_i for extreme values.

Figure 3.23b depicts the mean waiting time Σ for the *Size* algorithms for different file size thresholds T_s. We observe that for values smaller than the average image file size of $3.5\,\mathrm{MB}$, the *Size* algorithms performs similar to the *Immediate* algorithms. In this case the *Size* algorithms also triggers an upload for almost each file and shows the same behaviour as the *Immediate* algorithm. For size thresholds T_s smaller than $16\,\mathrm{MB}$, the performance of the *Size* algorithms is only slightly worse than the reference mechanism. Here, only a few files are required to trigger the upload process and the additional delay introduced by waiting is negligible. For larger size thresholds T_s, the mean waiting time Σ increases significantly.

Relative Disconnection Time

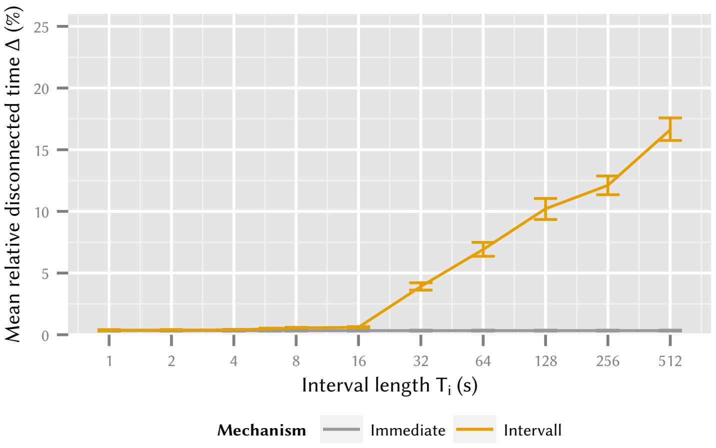
Besides a fast synchronisation, the users also demand a long battery life time of the mobile device. Besides the display, the radio interface used to establish

Internet connection is one of the main power consumers. In order to analyse the power savings of the different algorithms, we evaluate the relative disconnected time Δ during the synchronisation process. Therefore, we consider the time between the sending of the first and the last image of the mobile client and calculate the percentage during which no Internet connection is established. The mean relative disconnected times including the $95\,\%$ confidence intervals are depicted in Figure 3.24.

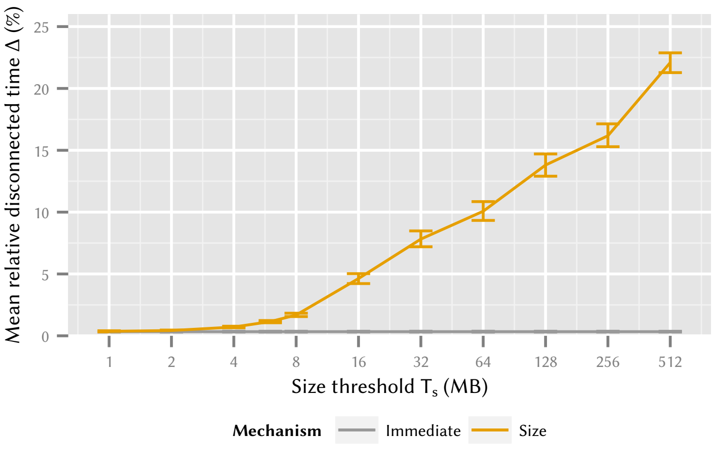
Figure 3.24a depicts the relative disconnected time Δ on the y-axis, on the x-axis the interval length T_i in s for the *Interval* algorithm. Figure 3.24b also depicts the relative disconnected time Δ on the y-axis, on the x-axis the size threshold T_s in MB for the *Size* algorithm. Both figures include the *Immediate* algorithm as a reference.

We first study Figure 3.24a. Similar to Figure 3.23a, we observe no significant differences between the *Interval* and *Immediate* algorithm for interval lengths T_i smaller than $8\,\mathrm{s}$, as both algorithms show the same behaviour here. For values larger than $8\,\mathrm{s}$, the *Interval* algorithm starts sending files in batches and no longer on a per file base. However, still no clear effect is visible, as the inter-sending interval and the mean image inter-arrival time of files are still smaller than the disconnection timeout of $11\,\mathrm{s}$. This results in an almost permanent Internet connection similar to the *Immediate* algorithm. For interval lengths T_i above $16\,\mathrm{s}$, the *Interval* algorithm starts saving connection time and the relative disconnected time Δ increases, resulting in additional power savings.

Figure 3.24b shows the relative disconnected time Δ for variable thresholds for the *Size* algorithm. We see again that the *Size* and *Immediate* algorithms do not differ for size thresholds T_s smaller than $4\,\mathrm{MB}$, similar to the results in Figure 3.23b. Afterwards, we observe an increase in saved relative connection time Δ for larger thresholds T_s, due to the fact that here files are sent in batches and the mobile client disconnects between the sending intervals, again enabling power saving potential for the mobile network client.



(a) Algorithm Interval, different interval lengths T_i



(b) Algorithm Size, different threshold sizes T_s

Figure 3.24: Comparison of algorithms with regard to relative disconnected time Δ.

Connection Count

After considering the requirements of the end-user, we have a closer look at the requirements of the mobile network operators. The network operator is mainly interested in minimising the signalling overhead introduced by connection establishment. Therefore, a minimisation of the number of connections K during the synchronisation process is desired. Figure 3.25 depicts the average number of connections K established during the synchronisation process, including the 95% confidence intervals. The number of connections is shown on the y-axis, the x-axis in Figure 3.25a shows the interval lengths T_i of the *Interval* algorithms, the x-axis in Figure 3.25b shows the *Size* algorithm size threshold T_s.

We observe a similar behaviour as in the previous evaluations, the average number of connection K for the *Interval* and the *Immediate* algorithms is the same in Figure 3.25a, if the interval lengths T_i are smaller then $4\,\mathrm{s}$. With increasing interval lengths T_i the connection count K also increases and reaches a maximum for an interval length of $32\,\mathrm{s}$.

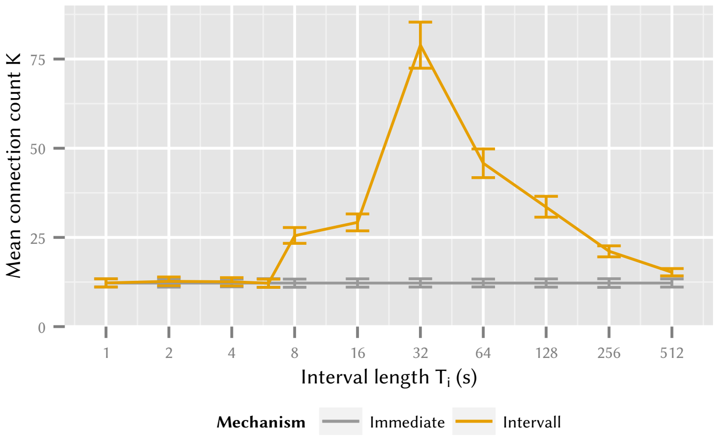
In order to explain this behaviour we make the following considerations. The maximum amount of data s_x which can be send from the camera to the mobile client during an intervals T_i of length x s is given by

$$s_x = x \cdot B_P = x \cdot 1/2\,\mathrm{MB}.$$

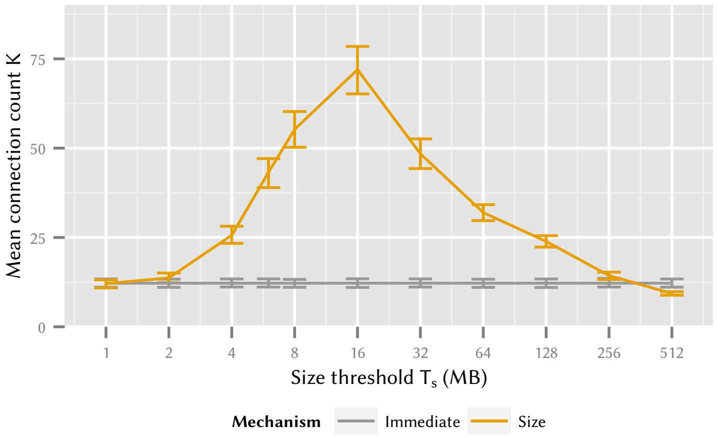
The average time t_x to upload s_x can now be calculated as

$$t_x = s_x / E\left[B_U\right] = x\ 0.5/8.0\,\mathrm{s} = x\ 1/16\,\mathrm{s}.$$

For interval lengths T_i between $8\,\mathrm{s}$ and $32\,\mathrm{s}$, t_x results in $0.5\,\mathrm{s}$, $1\,\mathrm{s}$, and $2\,\mathrm{s}$ respectively. Considering the disconnection timeout of $11\,\mathrm{s}$, we can see that a disconnect is likely for interval lengths T_i of $16\,\mathrm{s}$ and $32\,\mathrm{s}$. In order to explain the increased number of connections K for an interval length T_i of $8\,\mathrm{s}$, we also have to consider the average image file size of $3.5\,\mathrm{MB}$. Within $8\,\mathrm{s}$, a maximum of $s_x = 4\,\mathrm{MB}$ can be transferred from the camera to the mobile client. Therefore,



(a) Algorithm Interval, different interval lengths T_i



(b) Algorithm Size, different threshold sizes T_s

Figure 3.25: Comparison of algorithms with regard to connection count K.

it is likely that it takes two interval lengths T_i, respectively 16 s, to transfer the image. This explains the similar behaviour of the *Interval* algorithm for interval lengths T_i between 8 s and 16 s with regard to the connection count K.

For interval lengths T_i larger then 32 s, the connection count K decreases again. To explain this effect, we have to consider the maximum number of file batches transferred during the synchronisation process. If every file is transferred individually 1000 connections would be required, if all files are transferred in one single batch, only one connection would be established during the synchronisation process. Depending on the chosen interval lengths T_i, the average size of the batches varies, as larger intervals result in larger batches. The overall number of batches is limited, as we only consider a finite amount of 1000 images. Consequently, the maximum possible number of connections K is limited, too. However, this comes only into effect if large batches are used during the synchronisation process.

In Figure 3.25b we can observe similar behaviours of the *Size* algorithm as for the *Interval* algorithm in Figure 3.25a. For small thresholds, the *Size* and the *Immediate* algorithm result in the same number of connections K. With increasing sending thresholds, the number of disconnects and re-connections increases, as it takes longer to accumulate the required amount of new data at the mobile client. The maximum average connection count K is reached when using a 16 MB threshold, which corresponds to s_x for an inter-sending interval of 32 s. For larger thresholds, the maximum number of batches is the limiting factor for the connection count. This can especially be observed for very large thresholds. Considering a size threshold T_s of 128 MB, we can assume that almost every batch is transferred in an individual connection, as it takes $128\,\mathrm{MB}/B_P = 248\,\mathrm{s}$ to transfer the required data from the camera to the mobile client, but only $128\,\mathrm{MB}/E\left[B_U\right] = 16\,\mathrm{s}$ on average to upload the data from the mobile client to the cloud. Consequently, the transferred file size can be estimated as the product of the size threshold T_s and number of connections, i.e. $128\,\mathrm{MB} \cdot 25 = 3.2\,\mathrm{GB}$, which approximately matches the product of number of considered files and the average file size.

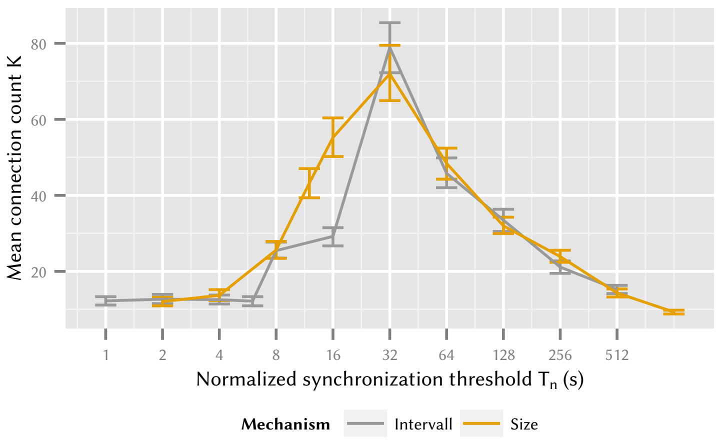


Figure 3.26: Comparison of algorithms with regard to normalized synchronization threshold T_n and connection count K.

Mechanism Comparison

The previous analyses imply that the results of the *Interval* and *Size* algorithm are interchangeable if the parameters are set appropriately. In order to test this hypothesis, we normalise the size threshold parameter T_s with the PAN bandwidth to calculate the effective average interval length caused by this threshold. The mean connection count K for both algorithms depending on the normalised synchronisation threshold T_n are depicted in Figure 3.26.

We observe that the mean connection count K of both algorithms are interchangeable most of the time, as the interval length T_i and the size threshold T_s can be converted into each other. However, the results for a normalised synchronisation threshold T_n of 16 s vary significantly. If we consider the *Interval* algorithm, the average amount of data transferred from the camera to the mobile client is $16\,\text{s} \cdot B_P = 8\,\text{MB}$, which is uploaded in approximately $8\,\text{MB}/E[U] = 1\,\text{s}$. For the *Size* algorithm, the amount of data transferred from the camera to the mobile client has to accumulate to $8\,\mathrm{MB}$ in order to result in a normalized synchronisation threshold T_n of $16\,\mathrm{s}$. Consequently, the mean upload time is also $1\,\mathrm{s}$. In conjunction with a disconnection timeout of $\tau = 11.576\,\mathrm{s}$ it is likely that the connection is closed after the upload in both cases.

However, even if the mean upload times in both cases are equal, the variance of the upload time distributions vary. The *Size* algorithm assures that always the same amount of data is uploaded, therefore the variance of the upload time distribution is only influenced by the variance of the upload bandwidth. In the case of the *Interval* algorithm, the amount of uploaded data varies, due to the different image sizes. Therefore, the variance of the upload time distribution in this case is dependent on the variance of the upload bandwidth and the variance of the image size. This increased variance leads in some cases to longer upload times avoiding a disconnect between two upload batches. However, this effect only comes into account if the sum of the upload time and the disconnection timeout is close to the inter-send interval.

The comparison of both algorithms indicate that similar results are reproducible with both of them. However, the time based algorithm allows a better control of the number of disconnects during the synchronisation process, as the inter-send interval can be adapted to the disconnection timeout. In order to adapt the size based algorithms accordingly, additional knowledge of the file arrival process is required to estimate the interval length T_i. In the following, we only consider the *Interval* algorithm as it is easier to configure and also more intuitive for the end-user than the *Size* algorithm.

Tradeoff Analysis for Considered Stakeholders

After analysing the different objectives of the stakeholders individually, we now consider the tradeoff between these contradicting optimisation goals, the mean waiting time for the file synchronisation Σ, the mean relative disconnected time Δ, and the mean connection count K.

Figure 3.27 depicts the mean waiting time for the file synchronisation on the

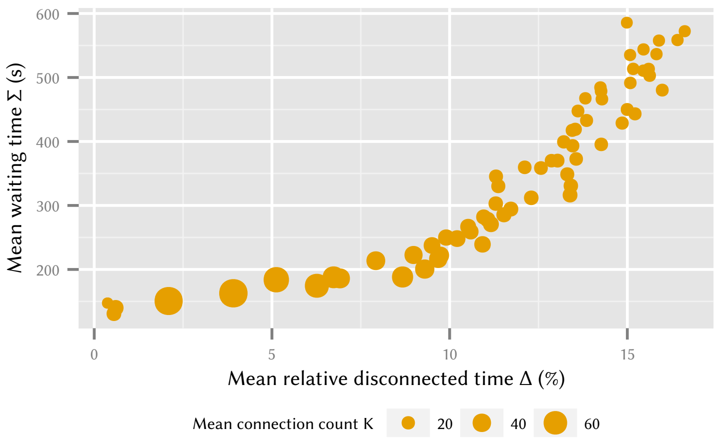


Figure 3.27: Trade-off analysis of identified stakeholder objectives.

y-axis and the mean relative disconnected time on the x-axis. Each point in the figure corresponds to one parameter setting for the *Interval*, the size of the point depicts the mean connection count K for this parameter setting. For each stakeholder objective we have different optimisation goals. While the user wants to minimise the mean waiting time Σ of the file synchronisation, the user also is interested in maximising the mean relative disconnected time Δ in order to save power. The network provider wants to minimise the signalling overhead, thus the mean connection count K should also be minimised. Therefore, an optimal parameter set would be located on the right bottom of the figure, small mean waiting time and high relative disconnected time Δ, and depicted by a small point indicating a small mean connection count K. However, the figure indicates that an increase in mean relative disconnected Δ time also results in an increased mean waiting time Σ. We see that allowing for a small increase in waiting time Σ of less than a minute can result in twice the time spent in disconnected state, i.e. additional power savings. This change in metrics can be facilitated by increasing the inter-send interval length from $32\,\mathrm{s}$ to $128\,\mathrm{s}$ for the *Interval* algorithm. An additional benefit of this change is a decrease of the connection count Δ of more than $50\,\%$, resulting in a significantly reduced signalling load in the network.

3.5 Lessons Learned

This chapter studied Internet *Video Streaming* and *File Synchronisation* as two prominent examples of modern network applications. During the operation and use of these applications, a number of stakeholders interact, each with different preferences and demands on what qualifies as optimal performance of the application. The *application provider* controls the application and directly influences the performance of the application for all other stakeholders. In case of the Video Streaming scenario, we consider the application provider to be interested in increasing the QoE for the application user while also maintaining a low utilisation of network and compute resources, as they would incur additional costs. In the File Synchronisation scenario the application operator is interested in the file synchronisation occurring as soon as possible, as this has been identified as a main impact factor of the users QoE. The *user* is interested in a high QoE as well as increasing the battery life of the UE. As in the last chapter, the *network operator* is interested to quantify and decrease the impact of application traffic on its network infrastructure.

The application operator has direct control over the application and is thus able to manipulate application behaviour in order to improve the respective key performance indicators. For the specific applications we consider adoption of the video transmission mechanisms and parameters for the video streaming scenario as well as diverse file upload scheduling mechanisms and parameters for the file synchronisation scenario. While users in these scenarios have no direct possibility to influence their key performance indicator, the video streaming scenario considers users to be heterogeneous, i.e. consisting of different subsets of users with different QoE requirements which have to be considered. Furthermore, we consider network operator to be passive, as Chapter 2 showed that it is not beneficial for them to optimise their network parameters for specific applications.

While studying both the Video Streaming and the File Synchronisation scenario, we find three major outcomes.

First, we study the impact of different video transmission mechanisms and parameter settings on energy consumption, number of connections established to the mobile network and traffic wastefully transmitted in case the user aborts video playback before completion. To this end, we provide both a network model for LTE mobile networks as well as playback models for the respective video transmission mechanisms. Then, we evaluate the models regarding the identified metrics for a range of relevant parameters. We find that live streaming the same content consumes at worst 711 % more energy than the download mechanism for the lowest possible bit rate. For the highest viable bit rate the ratio decreases, however the live streaming still consumes 117 % more energy than the download mechanism. In contrast, we observe that when considering the wasted traffic relative to the total content size, the download mechanism causes, even for the best case user scenario, 12 300 % of the data wasted by the live streaming scenario. The streaming mechanism results in only five times the wasted traffic of the amount lost during live streaming, while only consuming between 39.9 % and 89.3 % of the energy of the live mechanism. Overall, we find that the streaming mechanism provides good results for all considered metrics and can be customised by adapting the streaming buffer size to the application operators needs.

Next, we consider the impact of both the buffer size of the streaming mechanism and the available network load on different scenarios: Video Streaming or Video Browsing. We find that buffer sizes smaller than 0.5 s provide unacceptable QoE for all considered users regardless of sensitivity to stalling. Buffer sizes between 2 s and 4 s however are acceptable to most users regardless of sensitivity. When considering the Video Browsing scenario, we find that even if the length of the watched video is known QoE depends on the sensitivity of the user to initial stalling, however we are able to determine local QoE optima for all considered values of sensitivity.

Finally, when considering the file synchronisation scenario, we find that of the three considered scheduling mechanisms, both the interval and the size based algorithms, provide best results. If the application provider allows parameter selection for a given scheduling mechanism to be performed by the user, in order to increase QoE, the interval mechanism provides more intuitive configuration. This would for example allow the user to increase the inter-send interval from 32 s to 128 s which can double the time spent in disconnected state, saving energy and also reduce the connection count by 50 %, putting less strain on the network.

Based on the results obtained in this chapter, we observe that additionally to the tradeoffs in the network, highlighted in the last chapter, similar tradeoffs also exist in the application layer. Including the application providers as stakeholders and considering their key performance indicators requires new models but also allows us to better understand the impact of mechanisms implemented in applications. Key performance indicators of application providers sometimes overlap with those relevant to users, as application providers try to improve the experience of users in order to reduce churn.

Highlighted by both, the Video Streaming scenario and the File Synchronisation scenario, is the fact that oversimplifying the user population as one homogeneous group can decrease the overall QoE. While general QoE management mechanisms intent to optimise the MOS, this only increases the mean QoE, which may cause suboptimal results if the population of users present with a large variance in opinion scores. Thus, we suggest to either identify and cluster similar user groups or expose tradeoffs, within reason, to the user. This approach could be seen as extending the *Economic Traffic Management* approach to the user.

4 Resource Dimensioning and Management Schemes in Clouds

In this chapter we consider stakeholder tradeoffs in cloud scenarios. Traditionally, a cloud is a set of compute and network resources, which can be elastically rented by customers. With the recent rise of crowdsourcing platforms, also described as human-cloud, it has become useful to refer to this type of cloud as *machine cloud*, in order to better differentiate these two types of cloud. The human-cloud is based on the sample principles as the machine cloud, e.g. elasticity and reliability and enabled crowdsourcing employers to offer tasks to workers available on demand. In this chapter we consider multiple scenarios: First, we study the role of a cloud operator providing virtual resources to customers. Then, we consider decisions faced by a user of a cloud who is deploying virtualised network functions in a cloud. Finally, we investigate resource dimensioning in human-clouds. Regardless of the specific scenario, the number of resources available impact the KPIs of all participating stakeholders, and is thus subject to optimisation.

We first provide an overview of the involved stakeholders and KPIs in Figure 4.1. If we study the operation of a machine cloud, both the cloud operator and the cloud user need to be considered. The cloud operator is interested in increasing revenue, i.e. attracting a high number of customers, and decrease financial expenditure, e.g. by reducing consumed energy. The customer of a cloud operator is interested in good Service Level Agreements (SLAs), for example a low delay before processing of a job can begin. In the second scenario, the network function operator taking the role of a customer in the previous scenario,

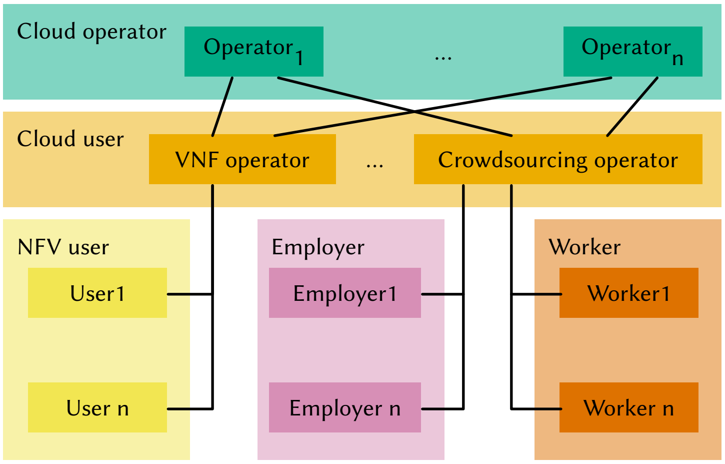


Figure 4.1: Stakeholders investigated in the cloud scenarios.

is interested in provisioning a minimal number of resources from the cloud operator, in order to reduce cost, while in turn providing a sufficient SLA to its customers. The users of the virtualised network functions demand a sufficient availability of the provided service. Finally, in case of the human-cloud scenario, the cloud operator has to satisfy two stakeholders with conflicting interests. On the one hand, employers are interested in a fast completion of the offered tasks. On the other hand, workers are interested in obtaining a high income. These goals are clearly conflicting, as fast completion can be obtained by providing a high number of available workers. However, income per workers increases if the tasks are distributed between fewer workers. Thus, the human-cloud operator has to balance the interests of the two stakeholders.

The contribution of this chapter is threefold

a) We provide a model for energy-efficient data centre operation and discuss sensible parameter configurations for the different stakeholders.

b) We study algorithms for resource provisioning on the example of a virtualised network function and evaluate their performance with respect to the demands of the stakeholders.

c) We model a crowdsourcing platform and provide guidelines for platform operators regarding resource acquisition.

The content of this chapter is published in [6, 10, 16]. In Section 4.1 we provide an overview of related work relevant to this chapter. Then, in Section 4.2 we discuss the tradeoffs faced by a cloud operator. We focus on the customer of a cloud operator in Section 4.3 and discuss challenges when provisioning virtualised network functions. Strategies for resource provisioning of a human-cloud are considered in Section 4.4. Finally, we provide lessons learned from our studies in Section 4.5.

4.1 Background and Related Work

This section provides related work relevant to this chapter. First, we discuss studies regarding energy efficiency in data centres in Section 4.2,. Then, we focus on research on mobile network traffic characteristics, to be used in Section 4.3. Finally, we focus on crowdsourcing research and provide background information for Section 4.4.

4.1.1 Energy-Efficiency in Data Centres

Several papers have been published, proposing new architectures for data centres which provide more resilience or are more cost effective[86–88].

Bolla et al. [89] provide an overview over approaches to reduce energy consumption caused by network infrastructure, offering a complementary view to the methods suggested in this chapter.

Heller et al. [90] published a paper considering the tradeoff between energy efficiency and resilience. They use the fat-tree architecture similar to [86, 87] which is based on commercial of-the-shelf network equipment. During normal network operation, the additional switches used for backup paths are switched off and only turned on in case of high load or network failures. The proposed mechanism is implemented in a testbed where OpenFlow is used for the switch management. However, they only turn off the switches and not the servers, which only consume between 5 % to 10 % of the overall energy consumption.

Kliazovich et al. [91] developed a simulation environment for computing the energy consumption of different data centre architectures. In addition to showing the share of network and server energy consumption, they present how much energy can be saved while using dynamic voltage and frequency scaling or dynamic power management.

One of the first paper presenting a dynamic resource management according to the offered load is presented by Chase et al. [92]. They propose an architecture where server clusters are dynamically resized in accordance to the negotiated SLAs.

A more detailed approach is presented by Chen et al. [93]. Three solutions are proposed to reduce the power consumption of servers in a data centre. For the first solution, the workload behaviour of the near future is predicted while the second solution is a reactive solution, using periodic feedback of system execution. The third proposed solution is a hybrid solution using a combination of prediction and periodic feedback.

The goal of the authors in [94–96] is to run a minimum number of servers in a data centre to maximise the revenue of the service provider. The considered data centre hosts a web page application. While in [94] the authors do not consider user impatience and the fact that servers consume energy without producing revenue during wake up, [96] takes both into account. In [95], the authors introduced a policy for dynamically adapting the number of running servers. The goal of the paper was to find the best tradeoff between consumed power and service quality.

In [97] the authors present a model for server farms using exponential interarrival, service and setup times. They consider different policies for powering down servers for finite and infinite servers.

4.1.2 Mobile Network Traffic Characteristics

The authors of [98] which include a co-author of [10], the basis for Section 4.3, provide a detailed evaluation of mobile network traces taken from a large European mobile network operator.

Having access to core network datasets, the authors of [99, 100] both take the approach of looking at high-level user traffic characteristics, focusing on temporal and spatial variations of user traffic volume and investigating the influence of different devices on this metric. Additional user flow and session traffic metrics are being studied in [101] with the conclusion that, in comparison to wired traffic, short flows are occurring more frequently. In 2006, a core network measurement study of various user traffic related patterns was conducted, providing an initial insight into Packet Data Protocol (PDP) context activity and durations [102].

In [103], mobile network traces are used to simulate a malicious signalling storm by transmitting low-volume user plane traffic with specially crafted inter-departure times, causing constant signalling. The authors of [104] investigate influence of core network elements on one-way delays in mobile networks.

4.1.3 Modeling Crowdsourcing Platforms

The term crowdsourcing is a neologism combining the terms ‘crowd’ and ‘outsourcing’. It was first introduced by Jeff Howe in 2006 [105] and describes a new form of work organisation with a smaller granularity than traditional forms [106]. In contrast to traditional forms of work organisation, work is divided in individual *tasks* that can be completed independent of each other. These tasks are not directly assigned to an employee but published on a *crowdsourcing platform* in form of an open call. Users publishing tasks on crowdsourcing platforms are referred to as *employers*, users accepting and accomplishing tasks as *workers*. Workers can freely choose which task to work on, other than in traditional forms of work organisation. In commercial applications, workers are usually paid for successfully completed tasks and do not receive hourly wages. Crowdsourcing platforms act as mediators in this environment, i.e., provide infrastructure for posting tasks and submitting task results and negotiate in case of disagreements between workers and employers.

The crowdsourcing approach is used for a large variety of non-profit, academic, and commercial applications, including information gathering during crisis [107], analysis of astronomic images [108], and by numerous large scale labour providers, e.g., Amazon Mechanical Turk (MTurk)[1], Microworkers[2], and Innocentive[3]. Depending on the specific use case, the features of the crowdsourcing platform, the workers, and employers differ. Therefore, we focus in this work on commercial micro-tasking platforms for the development and evaluation of our model.

Commercial micro-tasking platforms like MTurk or Microworkers are specialised labour providers for very fine granular tasks that can be easily completed by a human within a few seconds to a few minutes, but cannot be solved using automatic approaches. These tasks include, e.g., image tagging, text creation, or subjective ratings. As the tasks are highly repetitive, they are often submitted by the employers in form of *campaigns*, representing batches of similar tasks.

Several efforts have already been made to describe certain aspects of micro-tasking platforms in analytic models. Faradani et al. [109] modelled the arrival process of workers using a non-homogeneous Poisson process in order to derive optimal pricing strategies for the employers. The model is based on a crawled dataset from MTurk including about 130,000 campaigns with in total over 4,000,000 tasks. Wang et al. [110] analysed the completion time of crowdsourcing campaigns using a survival analysis based on a crawled dataset from MTurk consisting of more than 160,000 campaigns and approximately 6,700,000

[1]`http://www.mturk.com`, Accessed: November, 21st 2015
[2]`http://www.microworkers.com`, Accessed: November, 21st 2015
[3]`http://www.innocentive.com`, Accessed: November, 21st 2015

tasks over a period of 15 months. They were able to show the impact of time-independent factors, e.g., the payment or the type of the task, on the completion time. In order to optimise the costs and the completion times of single jobs, Bernstein et al. [111] use a $M/M/c$ queueing model to describe a crowdsourcing retainer approach. Here, workers are paid for staying online to wait for potentially upcoming tasks. The model was validated in a proof-of-concept experiment with 500 users on MTurk.

4.2 Data Centres

Data centres are used to host most of the applications running in the Internet, including those discussed earlier in this work. Servers are operated by the data centre operator and rented out to customers as part of a Platform as a Service (PaaS) or Infrastructure as a Service (IaaS) scheme. In order to increase revenue the data centre operator is interested in decreasing server power drain, one of the major matters of expense for data centres [112], while satisfying SLAs with their customers.

This section studies this scenario and provides a model intended for data centre operators to manage this tradeoff. In Section 4.2.1 we provide a mathematical formulation for the considered scenario. Then, in Section 4.2.2 we model this scenario using methods from queueing theory and derive metrics which can be used to evaluate the different approaches and configurations of data centres. In Section 4.2.3 we present different methods for solving the previously introduced queueing model. Finally, in Section 4.2.4 we study the performance implications of the model and discuss the tradeoff between power drain and suffered waiting time.

4.2.1 Considered Data Centre Architectures

A widely used data centre architecture is the three-tier architecture shown in Figure 4.2. The upper two layers of the architecture are responsible for distribut-

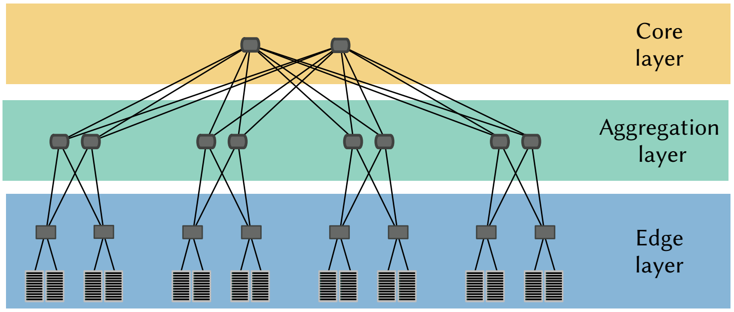


Figure 4.2: Considered standard three-tier data centre architecture.

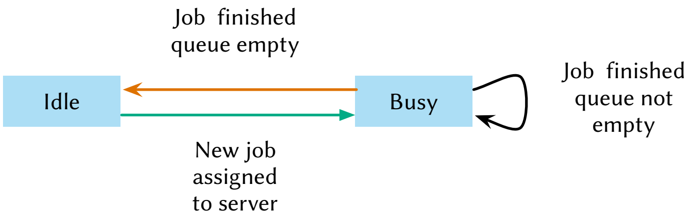
ing the traffic and consist of layer 3 switches where each switch has a backup switch. In this section, we focus on the edge layer and especially on a single Performance Optimised Data centre (POD). A POD consists of a number of servers connected over top of rack switches to an aggregation switch.

We assume that new jobs entering the system arrive with exponentially distributed inter-arrival time. When a job arrives at the POD, it is forwarded to an idle server. If no idle server is available, the job is queued. Once a server finishes processing its current job, it picks another one from the queue.

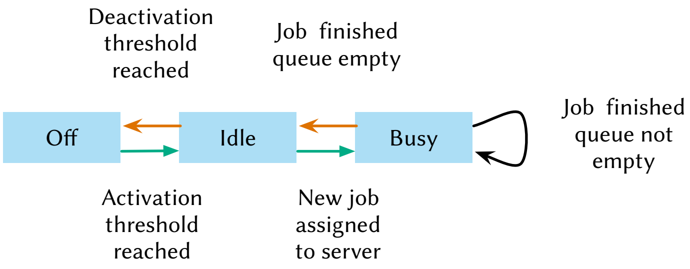
Our goal is to evaluate how much power is consumed in a data centre and how much can be saved when servers, currently not processing any job, are switched off. Therefore, we developed two different data centre models. The first model, the *traditional data centre*, consists of two-state servers only which are either *busy* or *idle*, as shown in Figure 4.3a For the second model, a more *energy-efficient data centre*, a subset of the servers may additionally be switched *on* and *off* on demand, shown in Figure 4.3b as recommended in [113].

Traditional Data Centre

For the traditional data centre model, each of the n servers is either on and processing a job or on and idle as depicted in Figure 4.3a. If a busy server finishes



(a) 2-state server model



(b) 3-state model of a reserved server

Figure 4.3: Assumed power state transition on a per server level.

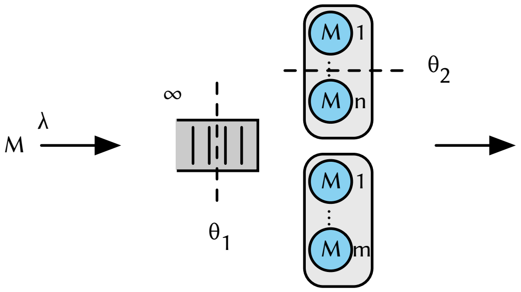


Figure 4.4: Considered system model for an energy-efficient data centre.

processing a job and the queue is empty, the server becomes idle. Once a new job is assigned to a yet idle server, the server becomes busy. According to our measurements of a server with an Intel twelve core processor 2.67 GHz and 32 GB RAM, a server currently processing a job consumes $e_{\text{busy}} = 240\,\text{W}$. An idle server still consumes $e_{\text{idle}} = 170\,\text{W}$.

Energy-Efficient Data Centre

For the second model, we differentiate between two types of servers: n base-line servers which are always on and m reserved servers to be enabled on demand. If they are enabled, their power drain is similar to that of the default data centre model. If they are disabled, each server consumes $e_{\text{off}} = 0\,\text{W}$. The n servers which are always enabled consume the same power as in the default data centre model. If the system queue has a length exceeding θ_2 where $\theta_2 \in (0, m)$ holds, the m reserved servers are enabled and stay enabled until the total number of jobs in the system drops to θ_1 for $\theta_1 \in (0, n)$. The transition between power levels for each of the reserved servers is depicted in Figure 4.3b.

The energy-efficient data centre operation model with the parameters θ_1 and θ_2 is depicted in Figure 4.4 and described in detail in the next section.

4.2.2 Model for Energy-Efficient Data Centres

In this section, we first discuss the default data centre model, where a server can either be idle or busy, processing a job. Afterwards, the energy-efficient data centre model with three states, i.e. idle, busy, or off, is defined.

Traditional Data Centre

We consider new jobs arriving at a POD with exponential iid inter-arrival times with rate λ. Each server accepts only one job at a time and processes it with an exponentially distributed service time with mean $\frac{1}{\mu}$. Then, the system can be modelled using a simple $M/M/n$ delay system. Here, the random variable X gives the number of jobs in the system and $x(i)$ is the stationary state probability that i jobs are currently in the system.

We obtain the mean power drain of such a system based on the measured values presented in Section 4.2.1. If less than $i < n$ jobs are currently in the system, then i servers are busy each consuming e_{busy} W and $n - i$ servers are idle, where each consumes e_{idle} W. If $i \geq n$ jobs are in the system all servers are busy and consume $n \cdot e_{\text{busy}}$ W in total.

For stationary state probabilities $x(i)$, we obtain the mean power drain as

$$E_{\text{max}} = \sum_{i=0}^{n} x(i)(ie_{\text{busy}} + (n-i)e_{\text{idle}}) + ne_{\text{busy}} \sum_{i=n+1}^{+\infty} x(i). \tag{4.1}$$

Furthermore, we can provide a lower bound for the power drain of the system by assuming that a server is turned off if it is not processing a job, thus consuming e_{off}. By substituting e_{off} for e_{idle} in Equation 4.1 we obtain

$$E_{\text{min}} = \sum_{i=0}^{n} x(i)(ie_{\text{busy}} + (n-i)e_{\text{off}}) + ne_{\text{busy}} \sum_{i=n+1}^{+\infty} x(i).$$

Energy Efficient Data Centre

We extend the queuing system to model the energy-efficient data centre model introduced in Section 4.2.1, by adapting the state space of the model. We now model the system state as a tuple (i, j) where i is gives the number of jobs in the system and j is 1 if the reserved servers are active and 0 if they are not.

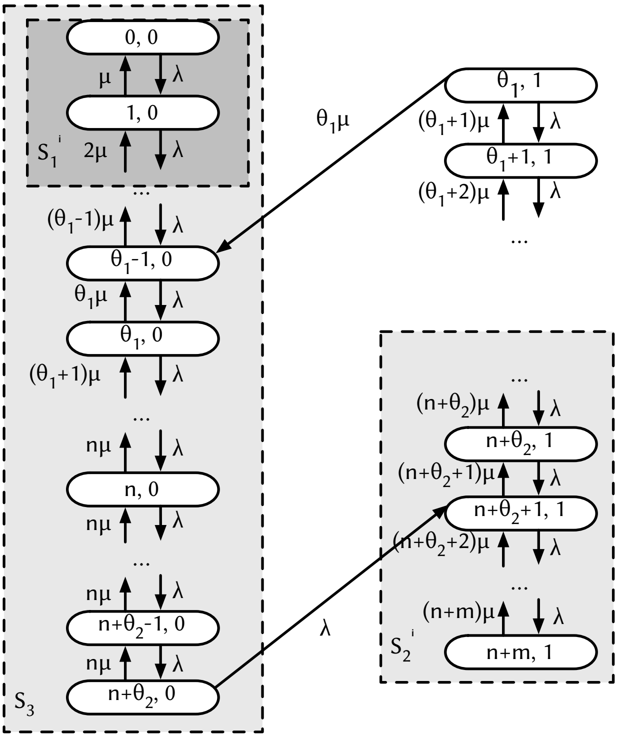


Figure 4.5: $M/M/(n+m)^{(\theta_1,\theta_2)}$ system with macro states S_1^i, S_2^i, and S_3 for the calculation of $x(i,\{0,1\})$ in the energy-efficient data centre.

The state diagram for this queueing model is given in Figure 4.5. The system activates the reserved servers if more than θ_2 jobs are in the queue, i.e. more than $n + \theta_2$ jobs are in the system. The reserved servers are deactivated if the number of jobs in the system drops below θ_1.

Again, X is the random variable describing the number of jobs in the system if the reserved servers are activated or deactivated, and $x(i, j)$ is the stationary probability that i jobs are in the system, and the reserved servers are activated, for $j = 1$, or deactivated, for $j = 0$.

Based on the state space and transitions, we formulate macro state equations, defined as the sum of all local balance equations of the states contained in the macro state. They provide, when solved, the state probabilities required for further analysis.

First, we consider the macro state equations for state S_1^i, c.f. Figure 4.5, which contains all system states where up to $i-1$ jobs are in the system and no reserved servers are activated. Depending on i, we obtain the following equations

$$i\mu x(i,0) = \lambda x(i-1,0) \qquad 0 < i < \theta_1, \tag{4.2}$$

$$i\mu x(i,0) + \theta_1 \mu x(\theta_1,1) = \lambda x(i-1,0) \qquad \theta_1 \le i \le n, \tag{4.3}$$

$$n\mu x(i,0) + \theta_1 \mu x(\theta_1,1) = \lambda x(i-1,0) \qquad n \le i \le n+\theta_2. \tag{4.4}$$

Next, we examine the system state if the reserved servers are activated. The macro state S_2^i contains all system states with activated reserved servers and at least $i + 1$ jobs in the system.

We get

$$\begin{aligned} i\mu x(i,1) = {} & \lambda x(i-1,1) \\ & + \lambda x(n+\theta_2,0) \qquad \theta_1 < i \le n+\theta_2+1, \end{aligned} \tag{4.5}$$

$$i\mu x(i,1) = \lambda x(i-1,1) \qquad n+\theta_2+1 < i \le n+m, \tag{4.6}$$

$$(n+m)\mu x(i,1) = \lambda x(i-1,1) \qquad n+m < i. \tag{4.7}$$

The third macro state S_3 contains all system states where only the base-line servers are activated and its equation states

$$\lambda x(n+\theta_2,0)=\theta_1\mu x(\theta_1,1). \tag{4.8}$$

Finally, the normalisation condition holds:

$$1=\sum_{i=0}^{n+\theta_2}x(i,0)+\sum_{i=\theta_1}^{+\infty}x(i,1). \tag{4.9}$$

Based on the state probabilities we can derive the required performance metrics for our analysis.

The carried traffic and utilisation is given by

$$a=\frac{\lambda}{\mu}\quad\text{and}\quad\rho=\frac{\lambda}{\mu(n+m)}.$$

Furthermore, we obtain the mean queue length

$$\Omega=\sum_{i=n}^{n+\theta_2}(i-n)x(i,0)+\sum_{i=n+m}^{+\infty}(i-(n+m))x(i,1).$$

By applying macro state Equation 4.7 we obtain for all $i>n+m$

$$x(i,1)=\rho x(i-1,1)=x(n+m,1)\rho^{i-(n+m)}, \tag{4.10}$$

Using this result and the first derivative of the geometric series we get

$$\begin{aligned}\Omega &= \sum_{i=n}^{n+\theta_2}(i-n)x(i,0)+x(n+m,1)\sum_{i=0}^{+\infty} i\rho^i \\ &= \sum_{i=n}^{n+\theta_2}(i-n)x(i,0)+x(n+m,1)\frac{\rho}{(1-\rho)^2}.\end{aligned}$$

Now, we can give the mean waiting time for all jobs in the system as

$$E[W]=\frac{\Omega}{\lambda}.$$

Finally, we obtain the mean power drain similarly to Equation 4.1 as

$$\begin{aligned}E=&\sum_{i=0}^{n}x(i,0)(ie_{\text{busy}}+(n-i)e_{\text{idle}}+me_{\text{off}})\\ &+\sum_{i=n+1}^{n+\theta_2}x(i,0)(ne_{\text{busy}}+me_{\text{off}})\\ &+\sum_{i=\theta_1}^{n+m}x(i,1)(ie_{\text{busy}}+(n+m-i)e_{\text{idle}})\\ &+x(i>n+m)(n+m)e_{\text{busy}}.\end{aligned}$$

4.2.3 Analysis of Proposed Data Centre Models

Using the macro state equations discussed in Section 4.2.2, there are multiple ways to obtain the state probabilities. This section examines the different approaches.

System of Linear Equations

First, it is possible to directly solve the system of linear equations implied by the micro or macro states. Solvers for linear equation systems scale cubic in the dimension of the matrix, which in this case is bounded by the system size. Especially for a large numbers of server, this prevents an exhaustive search of the parameter space.

Closed-form Solutions

Thus, we obtain closed form solutions for the state probabilities. These equations can be derived by recursively applying the macro state equations.

All equations feature a factor $x(0,0)$ which in turn can be calculated using the normalisation property given in Equation 4.9. Due to the length of the individual formulas, the following shorthand is introduced: For each state probability $x(i,j)$ depending on the factor $x(0,0)$ we define $\bar{x}(i,j) = x(i,j) \cdot x(0,0)^{-1}$, i.e. we cancel the factor.

For $0 < i < \theta_1$, we get

$$x(i,0) = x(0,0) \cdot \frac{a^i}{i!}.$$

As a further shorthand for substitution, we define

$$s_i = \sum_{k=0}^{i} a^k (n-k-1)!.$$

Using this definition, we get the state probability for θ_1 jobs in the system with activated reserved servers as

$$x(\theta_1,1) = x(0,0) \cdot \frac{a^{n+\theta_2+1}}{\left(1+\frac{a}{\theta_1}\right)} \cdot \frac{(\theta_1-1)!\left(\frac{(1-a^{\theta_2})\theta_1}{1-a} + a^{\theta_2} s_{n-\theta_1}\right)}{n^{\theta_2} n!\,(n-\theta_1+1)!}.$$

For $\theta_1 \leq i \leq n$, we get

$$x(i,0) = x(0,0) \cdot \left(\frac{\bar{x}(n,0)a^{i-\theta_1+1}}{(i-\theta_1+1)!} - \frac{\bar{x}(\theta_1,1)\theta_1 s_{i-\theta_1}}{i!} \right).$$

And for $n < i \leq n + \theta_2$, we get

$$x(i,0) = x(0,0) \cdot \left(\frac{\bar{x}(n,0)a^{i-\theta_1+1}}{n^{i-n}(n-\theta_1+1)!} \right. \\ \left. -\bar{x}(\theta_1,1) \left(\frac{\theta_1 s_{n-\theta_1} a^{i-n}}{n!} + \frac{\theta_1(1-a^{i-n})}{1-a} \right) \right).$$

Thus, we have all probabilities for system states where only the baseline servers are active. For the reserved servers, we obtain state probabilities for $\theta_1 < i \leq n + \theta_2 + 1$ as

$$x(i,1) = x(0,0) \cdot \left(\bar{x}(\theta_1,1) \frac{a^{i-\theta_1}\theta_1!}{i!} + \bar{x}(n+\theta_2,0) \sum_{k=1}^{i-\theta_1} \frac{a^k (i-k)!}{i!} \right).$$

For $n + \theta_2 + 1 < i \leq n + m$, we get

$$x(i,1) = x(0,0) \cdot \bar{x}(n+\theta_2+1,1) \cdot \frac{a^{i-(n+\theta_2+1)}(n+\theta_2+1)!}{i!},$$

and finally for $i > n + m$

$$x(i,1) = x(0,0) \cdot \bar{x}(n+m,1) \left(\frac{a}{n+m} \right)^{i-(n+m)}.$$

As discussed earlier, the probability of an empty system is given by the normalisation condition:

$$x(0,0) = \left(1 + \sum_{k=1}^{n+\theta_2} \bar{x}(k,0) + \sum_{k=\theta_1}^{\infty} \bar{x}(k,1)\right)^{-1}.$$

While these closed form solutions allow for the derivation of analytical properties of the model, performing a numerical analysis of a given parameter space is difficult due to numerical instability of the equations.

Recursive Algorithm

To avoid these problems, we introduce a recursive algorithm to calculate the state probabilities based on the macro state equations. To this end, we first define $x(0,0)$ as a constant K_0, and then iteratively compute the state probabilities. For an earlier application of this concept, see [114]. First, we calculate $x(i,0)$ for $0 < i < \theta_1$ as a factor of $x(0,0)$ using Equation 4.2. To obtain the probability for $x(\theta_1, 0)$, not only $x(\theta_1 - 1, 0)$, but $x(\theta_1, 1)$ is required, which we have not obtained yet. As this is the case with all $x(i,0)$ for $\theta_1 \leq i \leq n+\theta_2$ we implicitly introduce a second constant K_1 for $x(\theta_1, 1)$ and calculate all $x(i,0)$ for $\theta_1 \leq i \leq n + \theta_2$ as a linear combination of $x(\theta_1 - 1, 0)$ and K_1 as follows:

$$x(i,0) = x(\theta_1 - 1, 0)u_i + K_1 v_i. \tag{4.11}$$

For $i = \theta_1$, Equation 4.3 requires $u_{\theta_1} = \frac{a}{\theta_1}$ and $v_{\theta_1} = 1$. Continuing this pattern by successively applying Equation 4.3 for $\theta_1 < i \leq n$, we get

$$\begin{aligned} u_i &= \frac{a}{i} u_{i-1}, \\ v_i &= \frac{a}{i} v_{i-1} + \frac{\theta_1}{i}. \end{aligned}$$

We use Equation 4.4 to continue for $n < i \leq n + \theta_2$, and get

$$u_i = \frac{a}{n} u_{i-1},$$
$$v_i = \frac{a}{n} v_{i-1} + \frac{\theta_1}{n}.$$

Thus, we arrive at a probability for $x(n + \theta_2, 0)$ depending on $x(\theta_1 - 1, 0)$, which we have obtained, and $K_1 = x(\theta_1, 1)$ which we still need to acquire:

$$x(n + \theta_2, 0) = x(\theta_1 - 1, 0) u_{n+\theta_2} - K_1 v_{n+\theta_2}.$$

We apply Equation 4.8 and solve for K_1 and obtain

$$K_1 = \frac{\frac{a}{\theta_1} u_{n+\theta_2}}{\frac{a}{\theta_1} v_{n+\theta_2} + \theta_1} x(\theta_1 - 1, 0),$$

which allows to calculate the probabilities of $x(i, 0)$ for $\theta_1 \leq i \leq n + \theta_2$ using Equation 4.11.

We can now obtain the probabilities for states in which the reserved servers have been activated, beginning with $(\theta_1 + 1, 1)$ we apply Equation 4.5 for all $\theta_1 < i \leq n + \theta_2 + 1$ and get

$$x(i, 1) = \frac{a}{i} (x(i - 1, 1) + x(n + \theta_2, 0))$$

which we can calculate directly as all probabilities are known in relation to K_0. We continue applying Equation 4.6 and obtain

$$x(i, 1) = \frac{a}{i} (x(i - 1, 1) + x(n + \theta_2, 0)) \tag{4.12}$$

for $n + \theta_2 + 1 < i \leq n + m$.

Finally, we need to calculate the probability that the system is in states $(i, 1)$

for $i > n + m$ where we need to obtain

$$x(i > n + m, 1) = \sum_{i=n+m+1}^{+\infty} x(i, 1).$$

Due to the recursive definition of Equation 4.7, we can write

$$x(i, 1) = \rho x(i - 1, 1) = x(n + m, 1)\rho^{i-(n+m)}$$

for $\rho = \frac{a}{n+m}$ and $i > n + m$.

Applying this redefinition to Equation 4.12 we get

$$\begin{aligned} x(i > n + m, 1) &= \sum_{i=n+m+1}^{+\infty} x(i, 1) \\ &= x(n + m, 1) \sum_{i=1}^{+\infty} \rho^i. \end{aligned}$$

After applying the properties of the geometric series and basic transformations we get

$$x(i > n + m, 1) = x(n + m, 1)\frac{2 - \rho}{1 - \rho}.$$

Now that all probabilities are known in relation to K_0, we apply Equation 4.9 to obtain the inverse of K_1 and norm our values to obtain the real probabilities.

This approach allows for a fast and numerically stable calculation of the state probabilities and can be used to compute the required performance metrics for the complete parameter space.

4.2.4 Evaluation of Energy Saving Potential

Based on the metrics obtained in Section 4.2.2, we can now compare the introduced default data centre and energy-efficient data centre models.

An optimal system setting would decrease both waiting time and power drain. For the discussion of this optimisation problem, we require additional notation which is introduced first. We assume that the job inter-arrival rate λ, the job service rate μ, and the total number of servers n_{total} are constants and not subject to the optimisation process. Thus, the complete system can be described by the number of base-line servers n, the server activation threshold θ_2, and the server deactivation threshold θ_1. The number of reserved servers m can be easily derived if the total number of servers n_{total} is known. Given these parameters, we define $e(n, \theta_1, \theta_2)$ to be the mean power drain of the system and $w(n, \theta_1, \theta_2)$ be the mean waiting time of all jobs.

A general approach for solving such multi objective optimisation problems is defining a single aggregate objective function, such as:

$$f(n, \theta_1, \theta_2) = \alpha e(n, \theta_1, \theta_2) + (1 - \alpha) w(n, \theta_1, \theta_2), \tag{4.13}$$

for $0 \leq \alpha \leq 1$. Then, it is possible to choose an α in such a way that a desirable tradeoff is made. Thus, the optimisation problem can be defined as

$$\begin{aligned} \min f(n, \theta_1, \theta_2) \qquad s.t. \qquad\qquad 1 < n &< s, \\ 1 < \theta_1 &< n - 1, \\ 1 < \theta_2 &< m - 1, \end{aligned} \tag{4.14}$$

and trivially solved by evaluating all valid parameter combinations, sorting the objective function values and choosing the minimum.

This approach has the obvious disadvantage that while a parameter combination may be optimal according to the chosen objective function, it may very well not be optimal to the stakeholders in the scenario. For example, it may be possible that another system configuration exists with a minimally greater mean waiting time and a greatly reduced power drain. To be able to decide whether such a tradeoff exists, a more global view of the problem space is required. However, due to the number of possible parameter combinations, it is difficult to select appropriate parameters. Thus, we reduce the number of possible parameters by considering only Pareto-optimal states.

To define Pareto-optimality beyond the intuitive definition used earlier, we need to introduce the product order partial relation. Let $X \subseteq \mathbb{R}^n$ be our feature set. We set $x \preceq x^*$ for $x, x^* \in \mathbb{R}^n$ iff

$$x_i \leq x_i^* \qquad \forall 1 \leq i \leq n \tag{4.15}$$

holds. Then, x^* is Pareto-optimal in X if no $x \in X \setminus \{x^*\}$ exists, such that $x \preceq x^*$ holds.

To study the system behaviour we consider an exemplary rack of $n_{\text{total}} = 100$ servers, where new jobs arrive with a negative exponential inter-arrival time with mean $10\,\text{ms}$, yielding $\lambda = 1/10\,\text{ms}^{-1}$. To determine the mean service time we turn to [115] where it is reported that in average servers are operating at $10\,\%$ to $50\,\%$ of their maximum utilisation levels. With this in mind we assume that the service time for job completion is again negative exponential with a mean of $400\,\text{ms}$, which implies $\mu = 1/400\,\text{ms}^{-1}$, resulting in an overall utilisation of $\frac{\lambda}{\mu n_{\text{total}}} = 0.4$, well within the described limits.

Based on these parameters, we can compute the mean waiting time and power drain for the default data centre model. The mean waiting time achieved by the default data centre model provides a lower bound for the achievable waiting time for the energy-efficient data centre model, as all n servers are always either idle or busy. For the parameters described above, the default data centre model achieves a mean waiting time for all jobs of $E[W] = 4.75 \times 10^{-14}\,\text{ms}$. Furthermore, the mean power drain of the system, under the assumption that no servers are disabled, is set at $E_{\text{max}} = 100\,\%$, which provides an upper bound for the energy-efficient data centre model. However, if we assume that all servers are immediately switched off if they are not processing any jobs, we get $E_{\text{min}} = 48.48\,\%$, which is the lower bound for the energy-efficient data centre model.

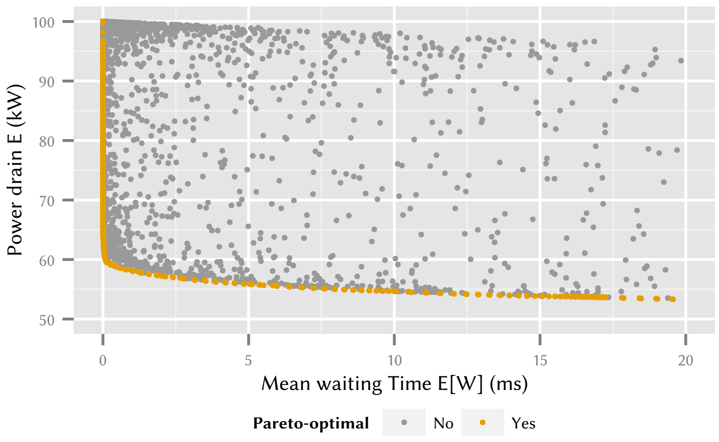


Figure 4.6: Set of Pareto-optimal values for the energy-efficient data centre.

Using the same parameters we evaluate the systems performance metrics, the mean power drain $e(n, \theta_1, \theta_2)$ relative to E_{max}, and the mean waiting time $w(n, \theta_1, \theta_2)$ for the energy-efficient data centre. As mentioned before, the Pareto-optima of the system are a subset of the $\mathbb{R}^2$, with one dimension corresponding to the mean waiting time, the other to the mean power drain.

We plot all Pareto-optima in Figure 4.6, the resulting curve has hyperbolic properties, going asymptotically to the mean waiting time $E[W]$ as well as asymptotically to a parallel of the lower bound of the power drain E. This allows us to select an acceptable increase in mean waiting time $E[W]$, for example one which would still satisfy a service level agreement, and harness the resulting energy savings. On the other hand, we can decide on the required energy savings and infer if the corresponding mean waiting times are acceptable. One possible parameter choice would allow the reduction of the power drain E by $40\,\%$ while only increasing the waiting time by less then one millisecond.

Given the set of Pareto-optima, we can investigate the parameter choice that

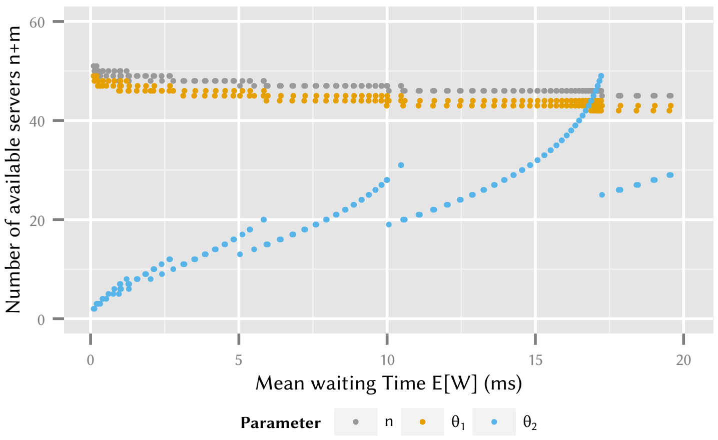


Figure 4.7: Impact of parameter selection on mean waiting time $E[W]$.

leads to these optima. To this end, we plot the system parameters for each optimum in Figure 4.7. The optima themselves are sorted according to the mean waiting time $E[W]$. From the figure we observe see that generally, the server deactivation threshold is very close to the number of base-line servers, in most cases $n - \theta_1 = 2$, the closest possible distance due to the macro state equation constraints. Furthermore, the number of base-line servers is decreasing as the mean waiting time $E[W]$ increases. The mean waiting time $E[W]$ spectrum can be partitioned in interleaving sections, during which the number of base-line servers remain constant. Furthermore, in such a section, the server activation threshold increases superlinearly.

4.3 Virtualised Network Functions in 3G Core Networks

In this section we apply the theoretical methods discussed in Section 4.2 and apply them to the real-world challenge of virtualised network functions. We consider the exemplary use case of a virtualised Gateway GPRS Support Node (GGSN). Here, network operators consider the virtualisation of previously physical middleboxes, in order to gain elasticity and reduce costs.

In contrast to the last section, we assume a loss model, as connection establishment requests are not queued in reality but expire if no capacity is available. Thus, we consider the blocking probability instead of the mean waiting time as a metric. As a second metric we consider the number of provisioned servers which need to kept powered on.

This section is structured as follows: In Section 4.3.1 we first introduce a model for a traditional GGSN. Then, we extend it to be applicable for the study of a virtualised GGSN. In Section 4.3.2 we describe the procedures used to obtain and process input parameters for use in our simulation study. Finally, in Section 4.3.3 we study possible gains by a virtualised GGSN by considering the tradeoff between the required servers to be active simultaneously and the incurred blocking probability.

4.3.1 Models of GGSN Implementations

In this section we provide a model for a traditional GGSN and discuss a model for a virtual GGSN using Virtualised Network Function (VNF). In VNF [116] static network middleboxes are replaced by commodity hardware. The tasks solved by the original middleboxes are then solved by dedicated software.

Traditional GGSN

First, we give a model for a *traditional* GGSN, i.e. a static network component. While we consider the GGSN to be one fixed entity, it can in reality consist of

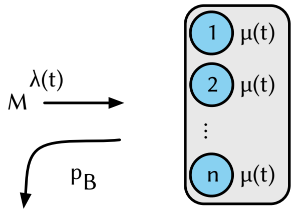


Figure 4.8: Considered model of a traditional GGSN.

multiple servers. However, due to the fact that the GGSN is purchased from a vendor as a middlebox, idle servers can be neither deactivated nor reused for other purposes.

We present an abstract queueing model for the traditional GGSN in Figure 4.8. New tunnels requests arrive according to a Poisson process with rate $\lambda(t)$ at the GGSN. This server will support a maximum tunnel capacity of c. When this capacity is reached, blocking will occur and newly incoming tunnels requests are rejected. Traditionally, GGSNs can be expected to be overdimensioned in such a way that this rarely happens. If the new tunnel is accepted, it will occupy one of the serving units of the server for the duration $\mu(t)$ of the tunnel. As stated earlier, we can not model the tunnel duration to be markovian, resulting in a $M/GI/c$ loss system. In order to give quality of service guarantees the network operator is interested in the system's blocking probability p_B, which we consider to be a key metric of our model. Additionally, the previously described diurnal patterns can also be modelled by adjusting the arrival and serving process distributions for each time of day. This alternatively also allows just to investigate the busy hour and thus the system's peak load.

GGSN using Network Function Virtualisation

Next, we introduce concepts from VNF, i.e. the idea to replace middleboxes with commodity hardware as an extended model in Figure 4.9. This allows us to re-

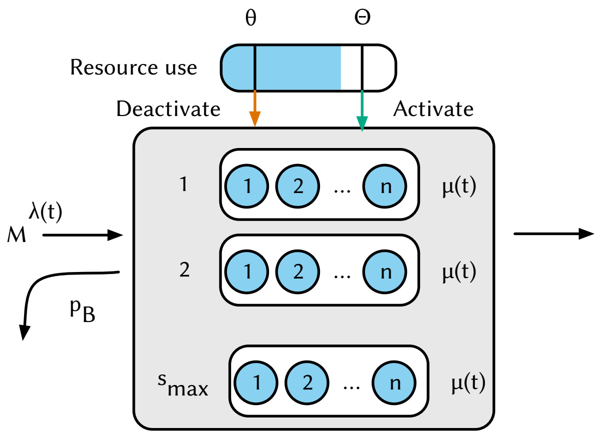


Figure 4.9: Considered model of a virtualised GGSN.

alise benefits from cloud computing, as we are now able to scale out, instead of up. The assumptions of the Markov arrival process $\lambda(t)$ and the serving time distributions $\mu(t)$ are carried over. However, instead of one server processing every tunnel, this model assumes that there are up to s_{max} virtualised servers s_i. Each of these is less powerful than the traditional GGSN, having a tunnel serving capacity of $c_i \ll c$ and a total system capacity of $c_{max} = s_{max} \times i$.

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

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

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

4.3.2 Mobile Network Traffic Characteristics

In order to evaluate our models introduced in Section 4.3.1, we use data gathered from a nation-wide mobile operator. This allows for precise core network evaluations and the creation statistical fits for the observed processes. In this section we first describe the dataset used for the evaluation and afterwards, we derive the random variables required for our models.

Dataset Description

All data was collected by the Measurement and Traffic Analysis in Wireless Networks (METAWIN) monitoring system [117] with measurement probes located at the Gn interface within the core network, , as shown in Figure 4.10. The Gn interface is a IP based Wide Area Network (WAN) used to connect GGSN and Serving GPRS Support Node (SGSN) installations. This access to the mobile core network provides METAWIN with a broad access to mobile signalling traffic.

For this investigation we employ GPRS Tunneling Protocol (GTP) protocol data gathered by METAWIN. This data includes the Radio Access Technology

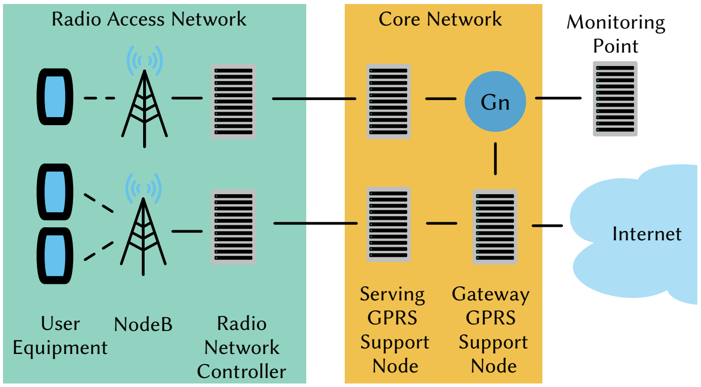


Figure 4.10: Overview of the METAWIN monitoring architecture in a 3G mobile network [117].

(RAT) identifier as well as the terminal types of the mobile clients, by use of the Type Allocation Code (TAC) part of the International Mobile Equipment Identity (IMEI). To meet privacy requirements, METAWIN anonymises all captured data. The application-level payload is removed and all user identifiers are hashed with one-way functions before data storage. Individual UEs in our dataset can be differentiated by the hashed Mobile Station Identifier (MS-ID), but not traced back to the actual user.

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

Statistical Evaluation

Using this dataset, we can obtain the distributions required for the models introduced in Section 4.3.1. First, we study the tunnel inter-arrival time in Figure 4.11.

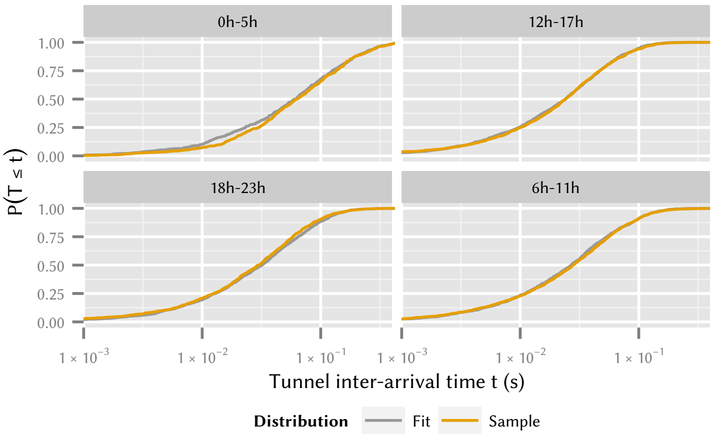


Figure 4.11: Empirical and exponentially fitted CDFs of the tunnel inter-arrival duration by time of day. CDFs are overlapping as the coefficient of determination is close to 1.

Table 4.1: Parameters for the exponentially distributed inter-arrival times and corresponding Pearson correlation coefficients.

Time of day	λ	R_{arr}
0h-5h	10.67	0.99
6h-11h	24.53	0.99
12h-17h	29.25	0.99
18h-23h	23.49	0.98

The arrival of new tunnel requests can be used as a measure for the load a GGSN experiences, as every incoming tunnel carries several signalling interactions, processing and state with it. Typically, a device will only hold one tunnel at a time, but this tunnel can be initiated and shut down in rapid succession, causing the aforementioned issues in the radio network. The arrivals also show a strong diurnal effect, closely resembling patterns present in the actual user traffic: We observe a decline of arrivals, i.e. longer inter-arrivals, late in the night and during the early morning hours with a peak rate in the afternoon and early evening. To represent this time-of-day dependence in the model, the measurement was split into the four time slots displayed in the figure. Each slot was then fitted with an exponential distribution by way of moment matching. This results in the cumulative negative exponential distribution function $F(x) = 1 - e^{-\lambda x}, x \geq 0$ with λ given in Table 4.1 for the four time slots. The fitted functions match the empirical data, with some deviation present at the left tail but overall with a positive correlation coefficient approaching 1.

The second important tunnel property is the duration of the PDP Context state accompanying a GTP tunnel held at the GGSN. Figure 4.12 shows the tunnel durations split up for the time of day, as there is once again a slight diurnal effect present, albeit with shifted peaks. Longer tunnels tend to occur at night, shorter tunnels during midday. Further properties of the tunnel duration, especially the correlation with device types and operating systems, are investigated in detail in [98].

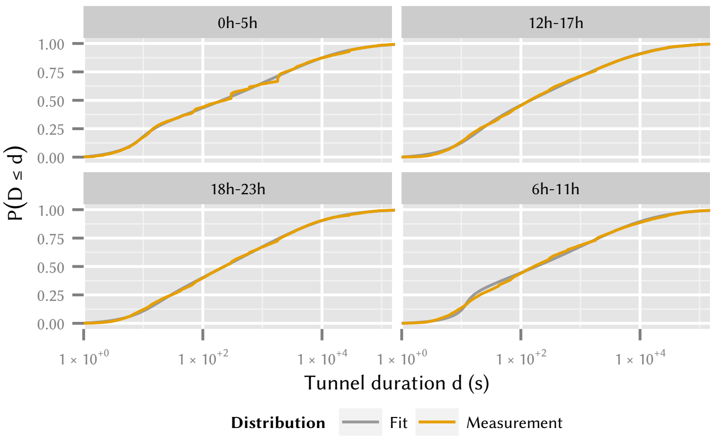


Figure 4.12: Empirical and fitted CDFs of the tunnel duration by time of day with fitted rational functions.

Table 4.2: Inverse functions fitted to the empirical duration distribution and correlation coefficients of the fit.

Time of day	Inverse fitted duration function	R_{dur}
0h-5h	$0.91 - 60.61y - 3498.78y^3 - \frac{110.70y+2289.94y^3}{y-1.00}$	0.99
6h-11h	$1 + 117.48y - 368.64y^2 - \frac{1720.13y^4}{y-1.00}$	0.99
12h-17h	$0.95 + 69.49y + \frac{81146.10y^3+1.08\times10^6y^5}{805-802.01y}$	0.99
18h-23h	$0.91 + 82.05y - \frac{2936.93y^4}{1.94y-1.95}$	0.99

Furthermore, the model requires information on the tunnel durations. However, none of the basic probability distributions, e.g. exponential, gamma, and Weibull distributions, fit the tunnel duration well enough. One of the reasons for this probably being the correlation of the tunnel duration to a large number of factors, including user behaviour and network-specific timers and procedures, e.g. tunnels are shut down by the network after specific events, introducing artefacts which make it hard to fit any distribution against. Instead, we fit rational functions to the empirical CDF using the Eureqa [118] software.

This allows for a much closer fit while still smoothing out some of the artefacts. Table 4.2 displays these functions fitted to the inverse CDF, to be directly used for generating random numbers using the inversion method. Both the CDF in Figure 4.12 as well as the Pearson correlation coefficient confirm the goodness of the fitted functions.

4.3.3 Comparison of Traditional and Virtualised Approach

We implement the models introduced in Section 4.3.1 using a Discrete Event Simulation (DES) with the SimPy[4] package as foundation. The implementation[5] as well as the considered scenarios[6] are also publicly available as a reference. To be in line with the measurement data we consider a simulation time for all simulation scenarios of 7 days, with a transient phase of 60 minutes. Ten replications of each scenario were performed. All error bars given in this section show the 5 % to 95 % quantiles of all replications.

We use the measurements introduced in Section 4.3.2 in order to dimension a traditional GGSN as a baseline for all further studies. Based on these results, we first examine the effects of network function virtualisation by scaling *out* instead of up through a virtual GGSN model. Finally, we arrive at a more realistic version of the virtual GGSN by taking the start-up and shut-down times into account.

[4] `https://simpy.readthedocs.org/`, Accessed: November, 21st 2015

[5] `https://github.com/fmetzger/ggsn-simulation/`, Accessed: November, 21st 2015

[6] `https://github.com/cschwartz/ggsn-simulation-studies/`, Accessed: November, 21st 2015

Traditional GGSN

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

In order to obtain a baseline dimensioning, we perform a simulation study, considering the impact of an increasing offered load on the blocking probability. We observe that as the number of supported parallel tunnels increases, the blocking probability decreases. For the normalized inter-arrival no blocking is occurring if we allow for more than 5000 parallel tunnels. Thus, we consider the range of 4000 to 5000 parallel tunnels to be of special interest for the remainder of the study.

Virtual GGSN

To study the feasibility of the virtual GGSN approach discussed in Section 4.3.1, we compare the performance metrics of the virtual GGSN with that of a traditional GGSN. To this end, the virtual GGSN is simulated in varying configurations. The number of servers and supported tunnels per server is chosen in such a way that the results can be compared with those obtained from our study of the traditional GGSN. Due to simulation time constraints, only a representative subset of scenarios is simulated.

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

Table 4.3: Manipulation check for the experimental factors based on one-way ANOVA.

	$F(2, 1275)$	η_p^2	p	Cohen's f^2	Cohen's $\hat{\omega}^2$
blocking probability p_B					
maxTunnels n	15601.53	0.993	< 0.001	26.73	0.96
maxInstances $S_{\max}$	10218.17	0.986	< 0.001	1.06	0.51
startstopDuration d	0.86	0.003	0.482	0.00	0.00
mean tunnel count n_A					
maxTunnels n	20448.34	0.994	< 0.001	27.71	0.96
maxInstances $S_{\max}$	13348.25	0.989	< 0.001	1.06	0.51
startstopDuration d	2.87	0.009	0.022	0.00	0.00

For the analysis of the influence of different model parameters on the performance metrics, we perform a one-way ANOVA with the results in Table 4.3. High values for the effect size estimators η_p^2 and Cohen's f^2[119] indicate that the main influence for both blocking probability p_B and mean number of tunnels n_A is the maximum number of tunnels n and virtual GGSN instances $S_{\max}$, i.e. the total number of possible concurrent tunnels in the system. Therefore, we study these parameters first.

In Figure 4.13 the CDF of the number of active servers for four different virtual GGSN configurations is displayed. We study the behaviour of a virtual GGSN with $S_{\max} = 30$ servers, where each server can support $n = 100$ or $n = 150$ tunnels. Then, we compare this with a virtual GGSN with $S_{\max} = 50$ servers and again $n = 75$ or $n = 150$ tunnels. We observe that increasing the number of supported tunnels n per server allows a larger percentage of servers to be shut-down or used for other tasks. This demonstrates the scaling capability of the virtualised model quite well. Note that both the scenario with 30 servers $S_{\max}$ and 150 maximum tunnels n per server as well as the scenario with 60 servers $S_{\max}$ and 75 maximum tunnels per server sharing the same maximum amount of tunnels, i.e. 4500, being right at the centre of the interesting range

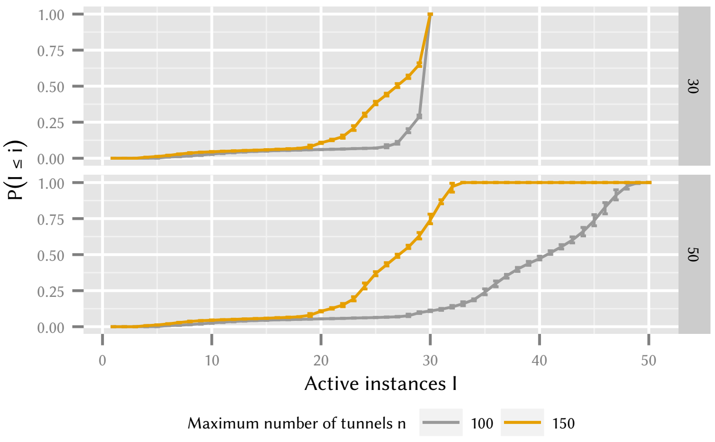


Figure 4.13: Impact of the maximum number of tunnels n and number of servers $S_{\max}$ on number of active servers in the virtual GGSN model.

of candidates.

Next, we study the blocking probability of the virtual GGSN system in Figure 4.14 and compare it to the results from the traditional GGSN model with both systems dimensioned for 4500 tunnels. We observe that, considering the start-up and shut-down time of 300 s, the blocking probability p_B increases by a factor of 1.46 if the virtual GGSN is comprised of 60 instances $S_{\max}$ dimensioned for 75 concurrent tunnels n , i.e. $\frac{1}{60}$ of the original server capacity. In this case 27 of all 60 servers can be turned off or used for other purposes at 50 % of the time. We conclude that choosing more powerful servers decreases the blocking probability but reduces the potential to disable servers.

So far we have considered a conservative start-up and shut-down time of servers d of 5 minutes, which can potentially occur in non-virtualised available hardware. In the next section we study the impact of reduced start-up and shut-down times with modern servers with fast storage, e.g. Solid State Disks (SSDs),

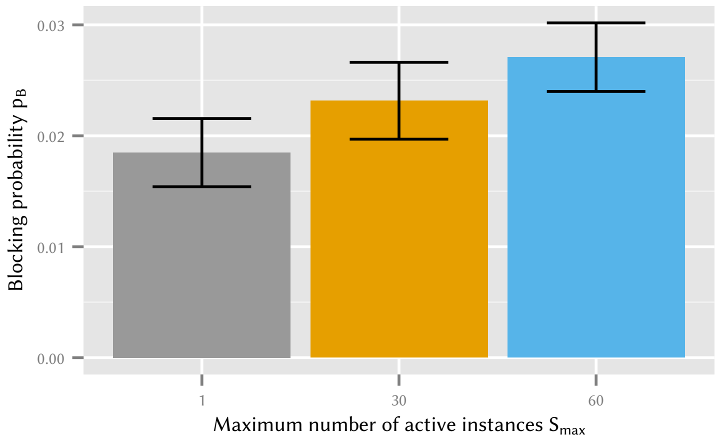


Figure 4.14: Impact of blocking probability p_B on the number of servers compared to the traditional GGSN, 4500 *maximum tunnels per server being on a single server, i.e.* 150 *on* 30*, and* 75 *on* 60 *servers.*

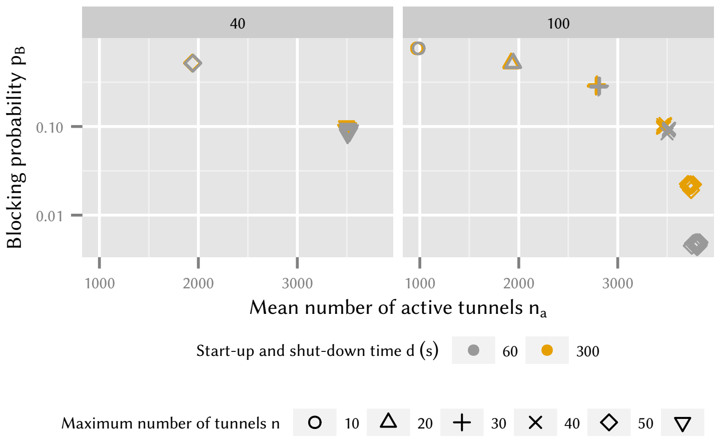


Figure 4.15: Trade-off between blocking probability p_B and mean resource utilisation n_A with regard to maximum number of instances $S_{\max}$, maximum number of tunnels per server n, and start-up and shut-down time d.

or containerised applications[7].

Impact of Start-up and Shut-down Times

In this section, we first consider the impact of different start-up and shut-down times d on resource utilisation n_A and blocking probabilities p_B. Afterwards, the influence of varying server start and stop times d on a fixed combination of maximum tunnels n and servers $S_{\max}$ in the system is examined.

Figure 4.15 shows scenarios with 40 and 100 GGSN instances $maxServers$ and 1000 to 5000 total concurrent tunnels. For each scenario, we study the impact of selecting a different maximum number of tunnels n per server as well as start-up and shut-down times d on blocking probability p_B and mean resource

[7] `https://www.docker.com/`, Accessed: November, 21st 2015

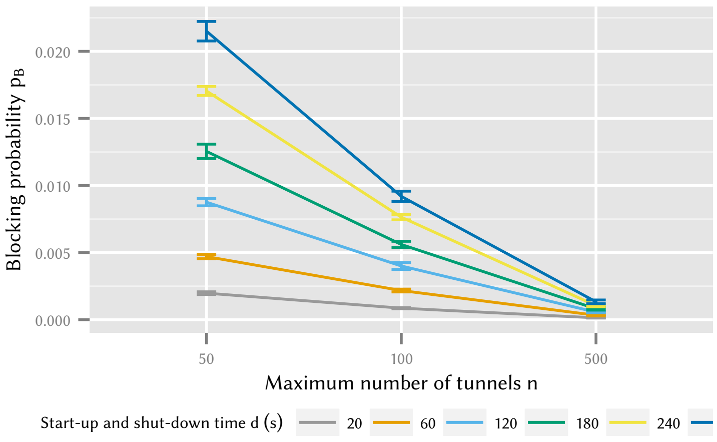


Figure 4.16: Influence of start-up and shut-down time d on blocking probability p_B with regard to different numbers of supported tunnels per instance n.

utilisation n_A. The first observation is that by increasing the number of servers $S_{\max}$, i.e. scaling out, the blocking probability p_B can be decreased, while maintaining a relatively low mean resource utilisation n_A. In addition to the previous effects, we notice that a higher start-up and shut-down time d causes a slight increase in blocking probability p_B for servers with low tunnel capacity n.

We focus on a specific scenario in Figure 4.16, where 5000 total tunnels should be supported by the system, to study this behaviour in more detail. To achieve this goal, we consider three types of instances, with the server capacity n varying between 50 and 500. In each case we change the start-up and shut-down time d between 20 and 300 s. We observe that lower server capacities n combined with higher start-up and shut-down times d increase the blocking probability p_B. This is due to the server start-up threshold mechanism, used in the model, not taking the additional capacity gained by activating an additional server into account. If a low capacity server with a long boot time is activated, there is a high probability that the system will quickly expend its capacity again.

Thus, it can be concluded that if smaller instances are to be used, e.g. due to the fact that they are cheaper than large instances, start-up and shut-down times should be kept minimal, for example by using containers or SSDs.

4.4 Dimensioning Crowdsourcing Platforms

While the last sections dealt with dimensioning of resources in machine cloud systems, similar methodologies are applicable to crowdsourcing, or human-clouds. Here, a crowdsourcing platform operator enables employers to distribute microtasks to workers. In order to ensure the success of the platform, the operator has to ensure the satisfaction of both the employers, as they provide the main source of income, as well as the workers, the resource of the platform. This tradeoff between employer satisfaction, i.e. time required before submitted tasks are completed, and worker satisfaction, i.e. income, has to be managed by the platform operator by carefully considering the number of workers employed at the platform.

To this end, in Section 4.4.1 we first provide a mode for crowdsourcing platforms regarding the two identified metrics. Then, we study parameters of a real world crowdsourcing platform in Section 4.4.2. Finally, in Section 4.4.3 we evaluate the provided model using the obtained parameters and discuss the tradeoff between employer and worker satisfaction from the point of view of the platform operator.

4.4.1 Considered Crowdsourcing Platform Model

In our model, schematically depicted in Figure 4.17, we consider a crowdsourcing platform employing c workers. The time between two campaigns being submitted is given by the random variable A_C Each campaign consists of a number of tasks, distributed according to the random variable Θ. We assume that each task is then completed by one of the c workers in order of arrival. The time

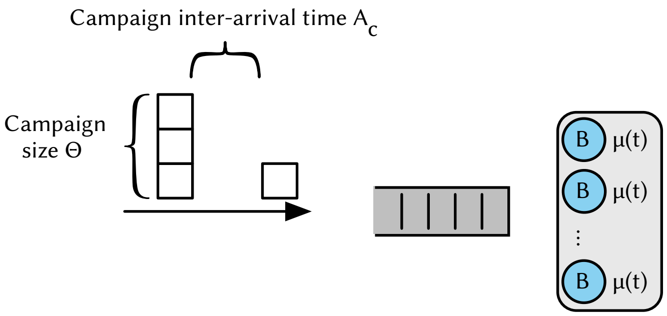


Figure 4.17: Considered crowdsourcing platform model.

required for completion is given by the random variable B.

From this model we derive two metrics in order to evaluate the performance of the crowdsourcing platform. First, we consider the utilisation ρ for all workers. This can be interpreted as a measure of earning potential for workers on the platform and should be maximized in order to keep current workers and attract new ones. Furthermore, we seek to obtain the mean task pre-processing delay $E[D]$, i.e., the time occurring before a worker begins to work on a task. This measure is relevant for the employer and should be minimised. We use the mean task pre-processing delay $E[D]$ instead of the average completion time of the campaigns, as the completion time also depends on the task length, which is under control of the employer and not of the platform operator.

In this section we first introduce an analytical model, which will be used to validate the simulation model discussed thereafter. Finally, a comparative validation of the analytical and simulative model is performed.

Analytical Consideration

First, in order to provide exact results, we consider the crowdsourcing platform as a $M^{[\Theta]}/M/c-\infty$ model. Here, we assume both the campaign inter-arrival time A_C as well as the time to complete a task B to be exponentially distributed with mean $E[A_C] = \frac{1}{\lambda}$ and $E[B] = {}^1/_\mu$, due to the large number of employers submitting tasks and the large number of workers completing them. Furthermore, we model the number of tasks per campaign Θ using a geometric distribution with mean $E[\Theta] = {}^1/_p$.

This model is well studied and state probabilities are provided in [120] or [121] for the case of a loss system.

Based on these state probabilities, we obtain the mean utilisation ρ per service unit as

$$\rho = \sum_{i=0}^{\kappa} \min(i, c)x(i) = \frac{\lambda E[\Theta]}{c\mu}.$$

This metric can be used to quantify the income of a worker, as a higher utilisation results in a higher income.

Next, we obtain the mean queue length Ω of the system as

$$\Omega = \sum_{i=c}^{\kappa} (i - c)x(i).$$

Now, we consider the mean task pre-processing delay $E[D]$ and with Little's theorem applied to the systems queue, we get

$$\lambda E[\Theta]E[D] = \Omega.$$

We solve for task the pre-processing delay $E[D]$ and obtain

$$E[D] = \frac{\Omega}{\lambda E[\Theta]}.$$

Detailed Simulation Model

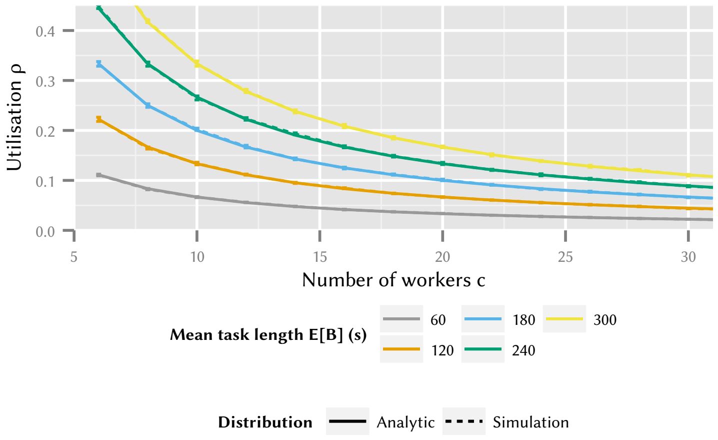
In order to allow for a larger variety of campaign inter-arrival time distributions A_C, we implement a discrete event simulation using the OMNet++ simulation framework[8]. We augment the framework with support for bulk arrivals and support of empiric distributions taken from measurements described in Section 4.4.2. Similarly to the queueing model introduced in this section, we consider campaign inter-arrivals according to a distribution A_C and a campaign size of Θ tasks. Task length is given by a distribution B and tasks are stored in an unbounded queue before being sent to service to the c available workers. During simulation, we record the mean utilisation ρ as well as the mean task pre-processing delay $E[D]$.

Impact of Campaign Arrival Process Type

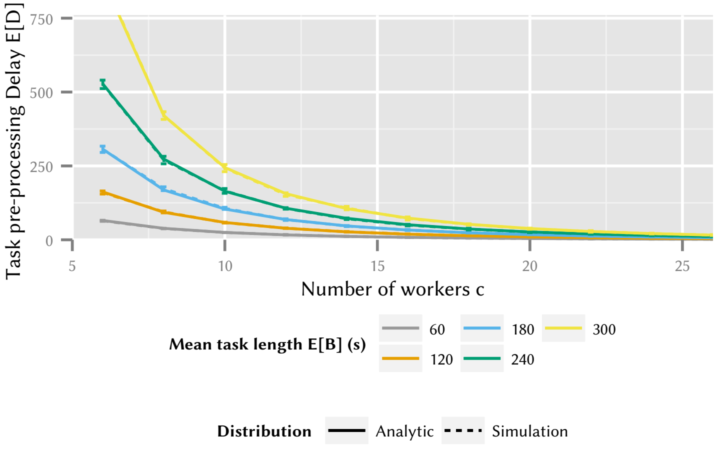
In this section, we validate the simulative model by comparing the metrics utilisation ρ and task pre-processing delay $E[D]$ for a representative parameter set with those obtained from the analytic model. We consider exponential campaign inter-arrival times A_C with a campaign rate of $4\,\mathrm{h}^{-1}$ campaigns, and a campaign size Θ geometrically distributed with a mean of 100 tasks per campaign. For the task length B, we consider a set of suitable values, to accommodate for different task types, between 60 s to 300 s per task. In both simulation and analytical model, we consider between 5 and 50 workers.

Results are shown in Figure 4.18. Simulative and analytical results, respectively, are shown by different line types. However, due to the good fit of the analytic and simulative model, the line showing the simulative results completely covers the analytic results. For the simulation we give 95 percent confidence intervals based on 10 replications. In this, and all following figures, confidence intervals are given as error bars. For each simulation we consider a simulation duration of 1500 h and accommodate for a transient phase of 150 h hours. We observe that for both the utilisation ρ and mean task pre-processing delays $E[D]$, the analytical results are well within the confidence intervals.

[8] `http://www.omnetpp.org/`, Accessed: November, 21st 2015



(a) Utilisation ρ



(b) Mean task pre-processing delay $E[D]$

Figure 4.18: Validation of simulation with analytic model.

4.4.2 Measurement of Platform Characteristics

In this section we analyse a large dataset from a commercial crowdsourcing platform to derive to derive realistic model parameters and compare the model based on these results with the analytic approximation.

Deriving Realistic Model Parameters

Our analysis is based on a large dataset from the commercial micro-tasking platform Microworkers.com. The dataset contains information about more than 160.000 campaigns submitted to the platform between May 2009 and Jan 2015, including the number of tasks per campaign as well as the time of the submission of the campaign.

Inter-arrival Times: First, we study the inter-arrival times of the campaigns. During the observation period, the platform faced some downtime due to software update or changes of the technical infrastructure. During this time, no campaigns could be submitted resulting in relatively large campaign inter-arrival times. In our model we only consider the regular operation of the platform, therefore we removed all inter-arrival times larger than $97.5\,\%$ quantile of all observed values, which affects about $2.5\,\%$ of all values.

Considering the remaining data, we observe a mean campaign inter-arrival time of 0.241 h with a standard deviation 0.346 h. Figure 4.19 shows the CDF of considered inter-arrival times, as well as the fitted distribution. For the fitting we considered several possible distributions but found the gamma distribution

$$P(A_c = t) \sim \Gamma(\alpha, \beta, t) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta t}$$

defined by shape α and rate β to be the most suitable. Using `fitdistrplus`[9] for the R language we derive the distribution parameters for the cam-

[9] `https://cran.r-project.org/web/packages/fitdistrplus`, Accessed: November, 21st 2015

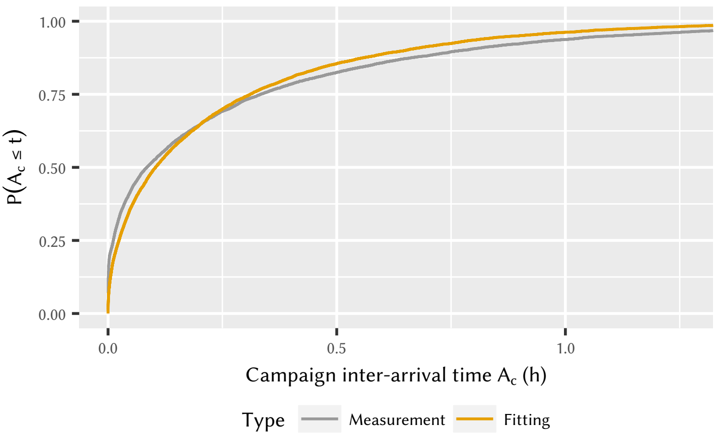


Figure 4.19: Observed campaign inter-arrival times A_c and corresponding fit.

paign inter-arrival times A_C by moment fitting and result in the estimated parameters $\alpha = 0.484$ and $\beta = 2.009$.

Campaign Sizes: Next, we consider the campaign sizes, respectively the number of tasks per campaign. The smallest possible campaign sizes on Microworkers is 30 tasks, however our dataset contained a few internal test campaigns with a small size.

These test campaigns, as well as outliers larger than the $97.5\,\%$ quantile of the campaign size have been removed from the considered dataset. In total $3.7\,\%$ of the original dataset were filtered by these conditions, the remaining data resulted in a mean campaign size of 97.01 tasks and a standard deviation of 103.41. The CDF of the campaign sizes Θ is depicted in Figure 4.20, together with the corresponding fitted distribution.

Due to the platform restrictions mentioned above, the campaign sizes start with a minimum value of 30 tasks. We observe that a very high share, i.e. $35\,\%$,

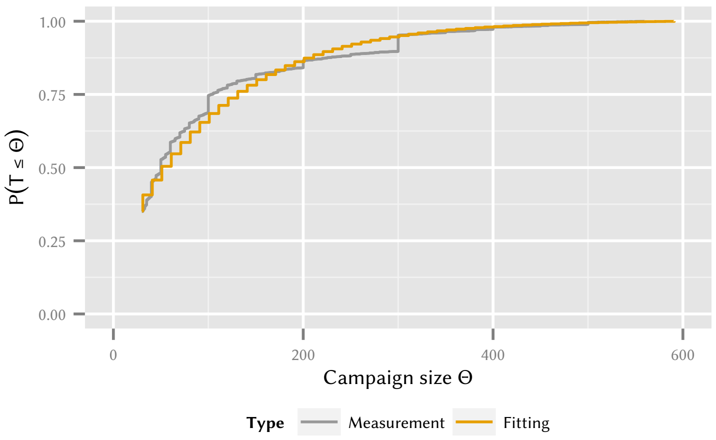


Figure 4.20: Observed campaign sizes Θ and corresponding fit.

of campaigns has only this minimum size. Further, campaign sizes which are a multiple of 10 or a multiple of 100 are quite frequent. This is caused by the fact that most tasks on Microworkers.com are repetitive and the employers choose the required number of repetitions and thus are more likely to round the number of repetitions to the nearest multiple of 10 or 100.

In order to obtain a suitable analytic distribution for the empiric values, we normalise the observed values and use the following piecewise defined distribution.

$$P(S = s) \sim \begin{cases} 0 & \text{if } s < s_{\min} \\ p_{s_{\min}} & \text{if } s = s_{\min} \\ GEOM(s) \cdot 10 + (s_{\min} + 1) & \text{else} \end{cases}$$

with

$$GEOM(s) = (1 - p)^s p.$$

The minimum campaign size $s_{\min}$ is observed with a fixed probability $p_{s_{\min}}$, while all campaign sizes larger than $s_{\min}$ follow a shifted and scaled geometric distribution.

Due to the relatively high frequencies of campaign sizes being multiples of 10 and 100, it is only possible to achieve a good fitting either for the lower or the higher region of the geometric part. As an overestimation of the campaign size will give us an upper bound of the platform work load, we decided to put a stronger emphasis on correct fitting of the larger campaign sizes.

We estimate the $p = 0.086$ parameter of the geometric distribution using quantile matching for the $90\,$percent quantile. The values $s_{\min} = 30$ and $p_{s_{\min}} = 0.350$ are obtained from the empirical values.

Task Duration: Another relevant model parameter is the length of the tasks B, i.e., the time a single worker needs to complete one task. Unfortunately, this information cannot be obtained from our dataset, as tracking of the individual workers is not possible. Therefore, we assume that the processing times follow a negative-exponential distribution, i.e.

$$P(t_p = t) \sim \mu e^{-\mu t}.$$

Even if the exact processing times are not available, each employer has to add an estimation about the time it takes to complete a task in the campaign description. In our dataset, $87.8\,\%$ of all tasks had an estimated completion time between $120\,\mathrm{s}$ to $300\,\mathrm{s}$. Therefore, we consider $\mu \in \{\frac{1}{300}, \frac{1}{240}, \dots, \frac{1}{120}\}\ \mathrm{s}^{-}1$ for the following evaluations.

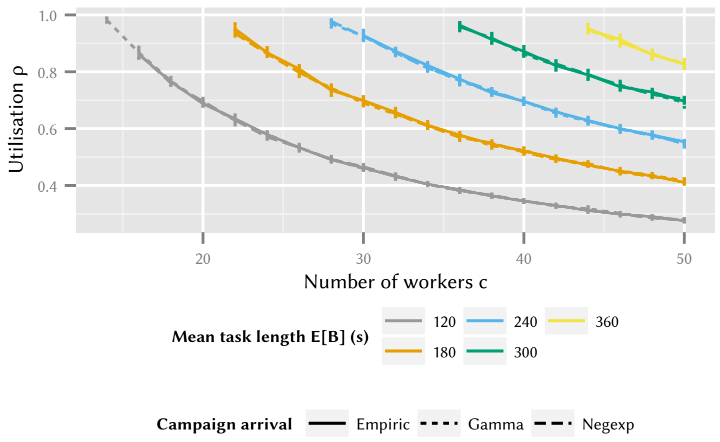
Number of Workers: Finally, the last model parameter to estimate is the number of users c on the crowdsourcing platform. At the time of this analysis, Microworkers.com had over 650.000 registered user accounts. However, this number is not applicable in the proposed model, for multiple reasons. The proposed model does not consider vacation times, i.e., the workers would have to be available 24/7. In reality, many crowdsourcing workers only work occasionally on the platforms or only for a few tasks. Further, employers can limit the access of to their campaigns to specific subsets of all workers, which is also not considered in the model. Moreover, Microworkers also limits the number of tasks a worker can complete in a single campaign. Taking this into account, the number of workers to be considered in our model has to be much smaller than the number of workers on the real world platform and consequently we decided to estimate meaningful values based on the model parameters instead of using the given number of workers from the dataset.

Comparison of Detailed and Analytical Model

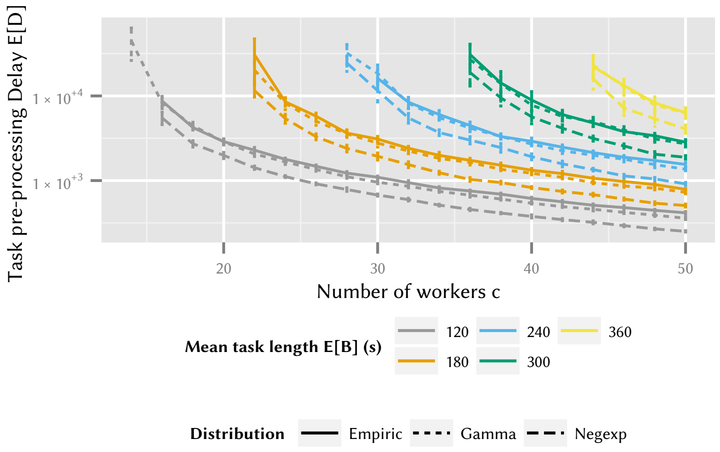
An important question for the later analysis is whether the analytic model from Section 4.4.1 can be used as an approximation or if a simulative evaluation is necessary. To this end we compared the later considered metrics utilisation ρ and task pre-processing delay $E[D]$ for *a*) a simulation using the empiric distributions for the task inter-arrival times and campaign sizes, *b*) a simulation using the fitted distributions derived earlier in this section, and *c*) the analytic model derived in Section 4.4.1. For the analytical model we used the campaign size distribution derived in this section and $\lambda = 4.14\,\mathrm{h}^{-1}$. The results of the different models are shown in Figure 4.21.

The utilisation ρ is depicted in Figure 4.21a, the task pre-processing delay $E[D]$ in Figure 4.21b. In both figures, the x-axis shows the number of workers c. The line colour indicates the mean task length, ranging from $120\,\mathrm{s}$ to $360\,\mathrm{s}$, the line style denotes the underlying model. We observe that all models result in the same utilisation ρ, which is not surprising when considering $\rho = \frac{E[A_c]E[\Theta]}{c\mu}$ with the mean inter-arrival time $E[A_c]$. Here, all parameters are the same for the three compared models and therefore, no significant differences can be seen.

This is different for the task pre-processing delay $E[D]$. Here, large discrepancies can been observed between the model based on the empiric distributions and the analytical model. This results show that the $M^{[\Theta]}/M/c - \infty$ model can also not be used as a worst case estimation, due to the fact that it underes-



(a) Utilization ρ



(b) Mean task pre-processing delay $E[D]$

Figure 4.21: Comparison of campaign arrival distributions.

timates the task pre-processing delay $E[D]$. In contrast to this, the simulation model based on the gamma distribution fits quite accurately the model based on the empirical values. Therefore, we continue our evaluation with the simulation model, based on the gamma distributed inter-arrival times and the piecewise defined distribution for the campaign sizes.

4.4.3 Evaluation of Platform Characteristics on Considered Metrics

In this section we use the simulative model introduced in Section 4.4.1 and the measurements obtained from the Microworkers platform in order to analyse the impact of different parameters on the considered metrics. First, we study the impact of campaign inter-arrival times. Then, we study tradeoffs between metrics of interest for the different stakeholders. The results presented in this section can be used as guidelines for platform operators, in order to ensure that both stakeholders are sufficiently satisfied.

Impact of Campaign Inter-arrival Distributions

Campaign inter-arrival times A_C influence both the work load of the individual workers ρ as well as the mean time required before a worker starts working on a task $E[D]$. From the perspective of an operator, understanding the influence of different inter-arrival processes is important. As shown in Section 4.4.2, the gamma distribution can be used to approximate the campaign inter-arrival times A_C as seen on the crowdsourcing platform Microworkers. In this section, we study the impact of such different processes by utilising the parameter space afforded by the gamma distribution and considering the impact on the metrics utilisation ρ and mean task pre-processing delay $E[D]$.

The characteristics of the gamma distribution change depending on the parameters shape α and rate β. While both shape and rate influence the mean

$$E[A_C] = \frac{\alpha}{\beta}$$

and variance

$$\mathrm{Var}[A_C] = \frac{\alpha}{\beta^2}$$

of the campaign inter-arrival times A_C, only the shape influences the skewness

$$\mathrm{Skew}[A_C] = \frac{2}{\sqrt{\alpha}}$$

of the distribution.

For a shape of $\alpha = 1$ the gamma distribution given by as

$$P(A_c = t) \sim \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta t}$$

degenerates to an exponential distribution with a PDF given as

$$a_c(t) = \beta e^{\beta t}$$

due to definition of the gamma function as $\Gamma(1) := 1$.

Increasing of the shape for the same rate changes the form of the distribution from an exponential type to a distribution which is similar to a normal distribution. By increasing the rate for the same shape the tightness of the distribution is modified. For rate parameters $\alpha < 1$ this results in a distribution with a long tail. The increase of the rate decreases the breadth of the distribution. Transferred to the campaign inter-arrival process A_C different shape and rate settings can be used to model different task types and varying the business of the platform. The range of the inter-arrival times is given by the rate and the shape defines the weighing of the different times. A lower shape means more campaigns arrive in bursts in combination with longer time periods without any campaign arrival.

Next, we use our simulation model introduced in Section 4.4.1 with the campaign size distribution Θ and task completion times B obtained in Section 4.4.2 for different campaign inter-arrival times to study the impact on the utilisation.

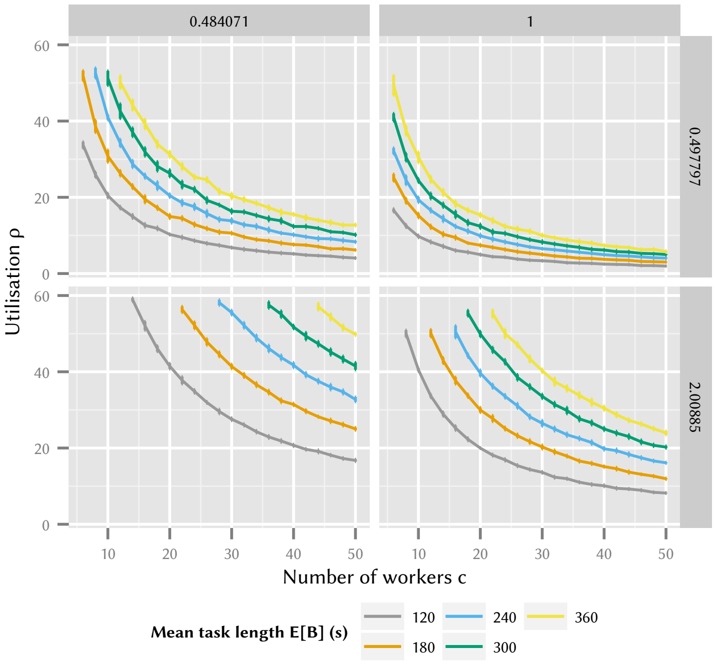


Figure 4.22: Utilisation ρ for different campaign inter-arrival times A_C.

Only stable systems, i.e., crowdsourcing platforms with a utilisation $\rho < 1$ are considered in the following.

Independent of the campaign inter-arrival time distribution A_C and the number of workers c, we see in Figure 4.22 that the introduction of more complex tasks in the platform by means of a higher mean task length $E[B]$ increases the utilisation ρ. The same number of workers c now require more time to process the same number of tasks. Furthermore, for the same campaign inter-arrival times A_C and mean task lengths $E[B]$, increasing the number of workers c decreases the utilisation ρ, as a higher number of workers has to compete for the same number of tasks. For different shapes α of the campaign inter-arrival times A_C and the same rate β, with all other parameters fixed, we observe a decrease of the shape results in an increase in utilisation ρ. A decrease of the shape α directly decreases the mean campaign inter-arrival time $E[A_C] = \frac{\alpha}{\beta}$ and increases the rate $\frac{1}{E[A_C]}$ of incoming campaigns, which increases the utilisation ρ. The same argument can be applied to the rate parameter of the campaign inter-arrival time distribution A_C. An increase of the rate β again influences the mean $E[A_C]$ and the rate of the campaign inter-arrival time A_C resulting in an increased utilisation ρ.

In Figure 4.23 we consider the impact of different campaign inter-arrival time characteristics A_C on the task pre-processing delay $E[D]$. For a fixed number of workers c and campaign inter-arrival distribution A_C a larger mean task duration B also increases the mean task pre-processing delay $E[D]$. As more tasks have to enter the queue, tasks which would not have been queued for lower task length now suffer queueing delay. For a fixed task length B and campaign inter-arrival distribution A_C, we see that increasing the number of workers c results in a decreased task pre-processing delay $E[D]$. The waiting probability decreases due to the higher capacity of the platform, resulting in a lower waiting time per task. Next, we consider the shape of the campaign inter-arrival time for fixed other parameters. The curves show that increasing the shape decreases the mean task pre-processing delay $E[D]$. This is caused by an increasing mean $E[A_C]$ of the inter-arrival times which results in a decrease of the campaign arrival rate. Thus, the platform contains fewer tasks for the same number of workers c and fewer tasks have to wait for completion. The effect is more obvious for higher traffic intensities.

Finally, we consider the effect of an increased rate β while keeping all other parameters fixed. An increased campaign inter-arrival rate increases the task pre-processing time $E[D]$, as the number of campaigns arriving at the platform is increased. The increase of the rate β decreases the variance of the campaign inter-arrival times distribution. For greater values of β, the mean campaign

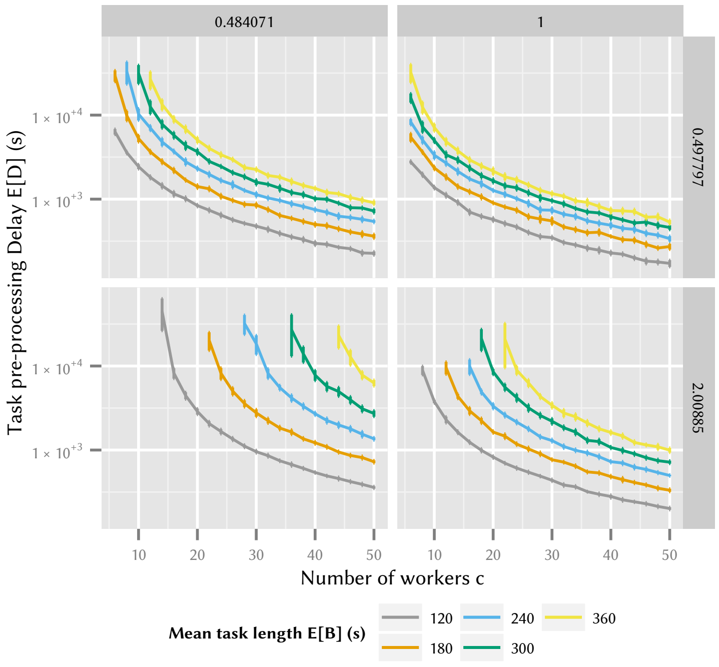


Figure 4.23: Mean task pre-processing delay $E[D]$ for different campaign inter-arrival times A_C.

inter-arrival time $E[A_C]$ decreases as the campaign inter-arrival rate increases. Thus, more tasks arriving at the platform and have to be completed with the same number of workers c.

Based on these observations, we conclude that while both shape and rate influence the metrics utilisation ρ and mean task pre-processing delay $E[D]$, the rate parameter β of the gamma distribution has a higher influence on the considered metrics. In order to account for the higher influence of the rate on the considered metrics, we fix the shape parameter α of the gamma distribution to the value 0.484071 obtained in Section 4.4.2 for the next section and focus on different rate parameters β.

Tradeoff Considerations for Platform Operators

A crowdsourcing platform operator's business success depends on the satisfaction of the main stakeholders, i.e., the employers and workers. As discussed in Section 4.4.1, workers are interested in a high utilisation ρ, due to the fact that this correlates with their payment. Employers are interested in having their tasks completed as fast as possible, i.e., in an as small as possible task pre-processing delays $E[D]$. The interests of the stakeholders are opposing as lower task pre-processing delays $E[D]$ can be achieved by hiring more workers, which in turn results in a lower utilisation ρ. Thus, the platform operator is forced to consider a tradeoff between worker and employer satisfaction, which we consider in this section. The impact of different campaign inter-arrival rates A_C on worker and employer satisfaction for the specific platform can be evaluated by following the coloured lines in Figure 4.24.

Given a fixed number of workers c, decreases in the campaign inter-arrival rate β result in lower utilisation ρ and longer mean task pre-processing delays $E[D]$. The effects on the utilisation ρ and the mean task pre-processing time $E[D]$ decrease for a larger amount of workers c. This means a platform with a larger number of workers is more robust against fluctuations in the rate of incoming campaigns β than a system with a small number of workers c.

Independent of the considered task duration B, we observe that increasing the

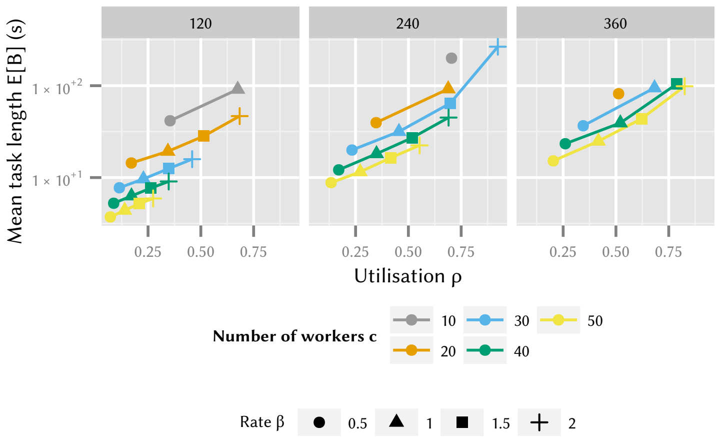


Figure 4.24: Tradeoff analysis between utilisation ρ and mean task pre-processing delay $E[D]$.

number of workers c, e.g. advertising the platform, decreases both the mean task pre-processing delay $E[D]$ as well as the utilisation ρ. However, this decrease is not linear. This means that a small increase of the number of workers c reduces the utilisation ρ, which is generally not desired. However, this small degradation of the utilisation ρ results in an over-proportional reduction of the mean task pre-processing delay $E[D]$. Thus, it is advisable to slightly overdimension the number of workers c to optimise the tradeoff between utilisation ρ and mean task pre-processing delay $E[D]$.

4.5 Lessons Learned

In this chapter we examined tradeoffs between stakeholders in cloud environments. As in the previously considered scenarios, various stakeholders exist, each with different and partially conflicting interests. First, we considered the operation of a data centre from the point of view of a *data centre operator*. The data centre operator is interested in decreasing expenditures, e.g. due to energy consumption of computing equipment as well as in offering a competitive service to its customers. The *data centre customers* are interested in obtaining such services, usually by selecting them according to metrics specified in a SLA. One example of a metric considered in a SLA is the delay a scheduled job is experiencing.

Second, we consider the role of the data centre customer in more detail. Thus, we focus on a specific virtualised network function deployed in a data centre, and the customer in the role of a *network function operator*. The network function operator is interested in reducing the number of concurrent virtual machines provisioned, in order to decrease cost. The network function operator in turn needs to satisfy its customers, the *network function users*, who rely on the network function operator satisfying availability goals, e.g. the ability to connect to the Internet using GTP tunnels.

Finally, we consider a human-cloud scenario. Here, the *crowdsourcing platform operator* attempts to balance the needs of its two stakeholders, the *crowdsourcing employer* against the requirements of the *crowdsourcing worker*. Crowdsourcing employers publish tasks via the crowdsourcing platform to crowdsourcing employees and require a fast task completion time. Crowdsourcing workers are interested in being offered as many tasks as possible, in order to increase their income. Crowdsourcing platform operators can dimension the number of available crowdsourcing workers in order balance this tradeoff.

We draw three major conclusions from this chapter:

First, we observe that the proposed scheme for operation of the data centre allows for a reduction of the energy consumption by 40 %. As a tradeoff, the time before a task can begin processing is increased by less then 1 ms. Furthermore, we show that for the considered mechanism, server deactivation should occur as soon as possible, resulting in the greatest energy savings while keeping an acceptable time until tasks can begin processing.

Second, we study the existence of configurations for the virtualised network function scenario, so that even for conservative server startup times, e.g. 300 s, the blocking probability increases only by a factor of 1.46. This configuration also allows 45 % of the required instances to be used for other purposes at 50 % of the time. We also demonstrate that the observed blocking probability can be reduced by over 90 % by employing techniques to reduce instance startup time, e.g. SSDs or software containerisation.

Finally, we show that according to our model, crowdsourcing platforms are robust regarding different shapes of the arrival process, i.e. bursty arrivals compared to periodic arrivals. Furthermore, we show that a relatively small number of workers is sufficient to sustain the platform during times of worker shortage, if the workers are put on retainer for the platform.

In the scenarios considered in this section, the platform operator is in control over parameters influencing the KPIs for the participating stakeholders. However, in all cases the KPIs of the other stakeholders, by means of SLA design, availability goals, or income, are also a KPI for the platform operator. This is due to the fact that not only one platform operator exists but multiple platform operators compete for customers. Research on such multi-operator scenarios can be performed using the models introduced in this chapter.

5 Conclusion

Today's Internet traffic is dominated by multiple stakeholders. Applications are developed and deployed by application providers, run on UEs produced by hardware vendors, and use mobile networks owned by operators. They use resources rented from cloud operators, may use human labour provided by crowdsourcing platforms and ultimately attempt to provide a high QoE to end users. However, the interests and KPIs of stakeholders in today's Internet do often collide with each other and sometimes even conflict.

For example, an application provider might be interested in providing its end users content as timely as possible using queries to a web service. These queries can result in numerous connection establishments and tear-downs, depending on the configuration used by the mobile network operator, increasing the signalling load in the mobile network and potentially causing *Signalling Storms*, i.e. overload. However, reconfiguration of the network by the operator can result in the UE being connected for a longer time, resulting in decreased battery life and QoE of the user.

In general, each stakeholder attempts to improve its considered KPIs by manipulating parameters under its control, e.g. by changing network configuration, implementing energy saving mechanisms, or adapting the number of available servers in a cloud environment. However, these manipulations not only improve the KPIs of the stakeholders but also impact the KPIs of a set of others. This results in complex relationships between stakeholders where interests are sometimes adverse and satisfactory results can only be reached by means of a tradeoff analysis.

In this monograph, we study clashes of interest for a set of scenarios from the major areas of the mobile Internet, including the network, the underlying application, and the cloud domain. We consider different approaches to model and analyse these conflicts and provide numeric results for best-case scenarios, which can usually be reached by cooperation between the participating stakeholders.

We begin with a study on the impact of a network's configuration on relevant KPIs. To this end we investigate the network traffic caused by mobile applications. Then, we examine the impact of application design by considering the impact of transmission mechanisms and scheduling algorithms implemented in mobile applications on KPIs for the participating stakeholders. Finally, we address the cloud by studying the impact of resource allocation and management schemes implemented in both machine-cloud and human-cloud scenarios.

In the first part we consider tradeoffs occurring in the network domain. We propose an algorithm to derive metrics such as power drain and signalling frequency from application traffic traces for a given network configuration. This algorithm allows application developers to consider the impact of their applications traffic on other considered stakeholders, i.e. on both the mobile network as well as the battery life of the UE. Then, we present an analytic model which allows the derivation of the considered metrics from arbitrary, theoretical traffic distributions. Using these methods we study exemplary application and perform a two-moment parameter study on synthetic traffic in order to identify problematic traffic patterns. We find that periodic traffic has a negative impact on both signalling frequency and power drain. We show that given the existence of proprietary fast dormancy algorithms, network timer optimisation performed by network operators can degrade performance for all participating stakeholders. Furthermore, it can result in equilibria with worse system performance for all participants compared to the case when no optimisation by the operator is performed. We suggest that hardware vendors implement operating system level mechanisms for applications to be notified on connection state changes in order to schedule transmissions and for network operators to provide interfaces to query network configuration.

In the second part we focus on the application domain by considering two specific applications: video streaming and cloud file synchronisation. We study different types of video transmission mechanisms and configurations regarding considered KPIs. While the configurable *Streaming* mechanism allows for suitable tradeoffs between all stakeholder pairs, we find that none of the considered transmissions mechanisms allows for suitable tradeoffs for all participating stakeholders. We suggest to use the *Design for Tussle* [77] in order to allow stakeholders to find suitable tradeoffs at run time. In order to study tradeoffs between end user groups with different viewing preferences, we study video QoE models in streaming scenarios and provide a model to evaluate consequences of parameter choice of the *Streaming* algorithm on user satisfaction. We show that by accounting for different user scenarios, i.e. browsing videos and watching videos, video QoE can be improved. Finally, we consider cloud file synchronisation services. Based on large scale measurements using the PlanetLab platform, we provide bandwidth and processing time distributions as well as a simulation framework to be used to evaluate different synchronisation scheduling algorithms. This simulation framework allows application developers to gauge the impact of their algorithm design decisions on other stakeholders, such as the network operator or the end user. We use the framework in order to evaluate different algorithms and find that both the *Interval* and the *Size* algorithms, allow for a good tradeoff between the considered stakeholders.

Having studied the application and network domains, we now focus on the cloud in the third part. We provide a queueing model as well as a power saving mechanism for data centres allowing operators to select a tradeoff between power savings they can achieve and SLAs they will be able to offer to their customers. We then consider the role of a cloud customer renting resources in a data centre in order to provide VNF services to mobile network operators. We propose and evaluate a resource provisioning mechanism allowing the VNF operator to balance the required resources with the SLAs which it can offer to its stakeholders. We especially consider the impact of new technologies, e.g. containerisation and SSDs on performance during provisioning. Finally, we apply our methodology to human-cloud scenarios and discuss dimensioning strategies for crowdsourcing platform operators, enabling them to provide a tradeoff between the interests of their stakeholders, the crowdsourcing platform employer and the crowdsourcing platform worker.

This monograph studies the impact of conflicts of interests between stakeholders in communication networks, where either of the stakeholders has the possibility to impact KPIs of other stakeholders or where stakeholders may choose between a set of competitors based on specific KPIs requirements. Methods and models introduced in this monograph can form the basis for further studies of other stakeholder interests which then can be analysed in a comparable way. Based on the results and proposed techniques multi-tier optimisation frameworks can be studied, investigating the clash of larger stakeholder groups over multiple scenarios.

Acronyms

CELL_DCH Dedicated CHannel. 13–17, 20–23, 31, 32, 34, 39, 40, 43, 45–49, 52, 54–56

CELL_FACH Forward Access CHannel. 13–17, 20–23, 39, 43, 47–49

RRC_Idle idle mode. 13–16, 20–23, 30, 31, 39, 40, 43, 45–49, 52, 54–56

3G Third Generation. vi, 11, 12, 17, 19, 29, 43, 51, 153, 155, 157, 159, 161, 163, 165, 167

3GPP 3rd Generation Partnership Project. 2, 16, 20

CDF Cumulative Density Function. 26, 158, 160, 161, 163, 173, 174

CN Core Network. 11, 12

CRX Continuous Reception. 73

DASH Dynamic Adaptive Streaming over HTTP. 64

DES Discrete Event Simulation. 5, 7, 161

DRX Discontinuous Reception. 71, 72, 74

GGSN Gateway GPRS Support Node. vi, 4, 8, 153–157, 159, 161–166

GTP GPRS Tunneling Protocol. 156, 157, 159, 186

HAS	HTTP Adaptive Streaming. 64, 65
HTTP	Hyper Text Transfer Protocol. vi, 64–66, 88, 95, 97
HTTPS	HTTP Over TLS. 25
IaaS	Infrastructure as a Service. 2, 135
iid	independent and identically distributed. 7, 44, 139
IMEI	International Mobile Equipment Identity. 157
IP	Internet Protocol. 19–21, 156
ISP	Internet Service Provider. 80, 84, 88
KPI	Key Performance Indicator. 1, 2, 4, 6, 8–10, 17, 18, 57, 58, 73, 129, 187, 189–192
LTE	Long Term Evolution. vi, 17, 43, 64, 67, 68, 71, 72, 77, 88, 105, 106, 126
M2M	Machine to Machine. 11
METAWIN	Measurement and Traffic Analysis in Wireless Networks. 156, 157
MOS	Mean Opinion Score. 23, 39, 40, 42, 66, 68, 88, 95, 127
MS-ID	Mobile Station Identifier. 157
P2P	Peer to Peer. 66
PaaS	Platform as a Service. 2, 135
PAN	Personal Area Network. 105, 106, 122
PD	Power drain. 23, 31–37, 49, 50, 55–57

PDCCH Physical Downlink Control Channel. 71, 73

PDF Probability Density Function. 75, 180

PDP Packet Data Protocol. 133, 159

POD Performance Optimised Data centre. 136, 139

PSTN Public Switched Telephone Network. 12

QoE Quality of Experience. v, vi, 2, 4, 6, 7, 9, 11, 17, 23, 26, 29, 37, 38, 40–42, 57, 62–68, 71, 88, 95–99, 101–103, 125–127, 189, 191

QoS Quality of Service. 12, 65, 66, 88

RAN Radio Access Network. 11, 12, 17, 29, 43

RAT Radio Access Technology. 156

RNC Radio Network Controller. 12, 14, 22, 29, 31, 32

RRC Radio Resource Control. v, 11–23, 43, 44, 47, 49, 50, 55, 57, 58, 71, 72

RTSP Real-Time Streaming Protocol. 64

SDN Software-Defined Networking. 66

SF Signalling frequency. 22, 23, 29–37

SGSN Serving GPRS Support Node. 156

SI Signalling intensity. 48–50, 52–54, 57

SLA Service Level Agreement. 4, 7, 129, 130, 135, 186, 187, 191

SOS Standard deviation of Opinion Scores. 66

SSD Solid State Disk. 164, 168, 187, 191

TAC Type Allocation Code. 157

TCP Transmission Control Protocol. 50, 65

UE User Equipment. 2, 4, 7, 11–20, 22, 23, 25, 29–31, 34, 39, 40, 43–45, 47–49, 52, 55–60, 68, 71, 73, 74, 76, 77, 80, 83–85, 88, 105, 106, 108, 125, 157, 189, 190

UMTS Universal Mobile Telecommunications System. v, 11, 12, 16, 43, 49

VNF Virtualised Network Function. 4, 153, 154, 191

VoIP Voice over IP. 25

WAN Wide Area Network. 156

Bibliography and References

Bibliography of the Author

Book Chapters

[1] T. Zseby, T. Zinner, K. Tutschku, Y. Shavitt, P. Tran-Gia, C. Schwartz, A. Rafetseder, C. Henke, and C. Schmoll, "Multipath Routing Slice Experiments in Federated Testbeds", in *Future Internet Assembly (FIA) Book Future Internet: Achievements and Promising Technology*, J. Domingue, A. Galis, A. Gavras, T. Zahariadis, D. Lambert, F. Cleary, P. Daras, S. Krco, H. Müller, and M.-S. Li, Eds., Springer, May 2011.

Journal Papers

[2] N. P. Ngoc, T. N. Huu, T. V. Quang, V. T. Hoang, H. T. Thu, P. Tran-Gia, and C. Schwartz, "A New Power Profiling Method and Power Scaling Mechanism for Energy-aware NetFPGA Gigabit Router", *Computer Networks*, vol. Special Issue: Green Communications, 2014.

[3] H.-T. Nguyen, N. P. Ngoc, T.-H. Truong, T. T. Ngoc, D. N. Minh, V. G. Nguyen, T.-H. Nguyen, T. N. Quynh, D. Hock, and C. Schwartz, "Modeling and Experimenting Combined Smart Sleep and Power Scaling Algorithms in Energy-aware Data Center Networks", *Simulation Modelling Practice and Theory*, vol. 39, Dec. 2013.

[4] C. Schwartz, T. Hossfeld, F. Lehrieder, and P. Tran-Gia, "Angry Apps: The Impact of Network Timer Selection on Power Consumption, Signalling Load, and Web QoE", *Journal of Computer Networks and Communications*, vol. 2013, Sep. 2013.

[5] D. Hock, M. Hartmann, C. Schwartz, and M. Menth, "ResiLyzer: Ein Werkzeug zur Analyse der Ausfallsicherheit in Paketvermittelten Kommunikationsnetzen", *PIK - Praxis der Informationsverarbeitung und Kommunikation*, vol. 34, Aug. 2011.

Conference Papers

[6] C. Schwartz, K. Borchert, M. Hirth, and P. Tran-Gia, "Modeling Crowdsourcing Platforms to Enable Workforce Dimensioning", in *International Telecommunication Networks and Applications Conference*, Sydney, Australia, Nov. 2015.

[7] L. Dinh-Xuan, C. Schwartz, M. Hirth, F. Wamser, and H. T. Thu, "Analyzing the Impact of Delay and Packet Loss on Google Docs", in *7th International Conference on Mobile Networks and Management*, Santander, Spain, Sep. 2015.

[8] S. Gebert, C. Schwartz, T. Zinner, and P. Tran-Gia, "Continuously Delivering Your Network (Short Paper)", in *IEEE/IFIP International Symposium on Integrated Network Management (IM)*, Ottawa, Canada, May 2015.

[9] T. Hoßfeld, C. Moldovan, and C. Schwartz, "To Each According to his Needs: Dimensioning Video Buffer for Specific User Profiles and Behavior", in *IFIP/IEEE International Workshop on Quality of Experience Centric Management (QCMan)*, Ottawa, Canada, May 2015.

[10] F. Metzger, C. Schwartz, and T. Hoßfeld, "GTP-based Load Model and Virtualization Gain for a Mobile Network's GGSN", in *5th International Conference on Communications and Electronics*, Da Nang, Vietnam, Jul. 2014.

[11] C. Schwartz, M. Hirth, T. Hoßfeld, and P. Tran-Gia, “Performance Model for Waiting Times in Cloud File Synchronization Services”, in *26th International Teletraffic Congress (ITC)*, Karlskrona, Sweden, Sep. 2014.

[12] M. Hirth, S. Scheuring, T. Hoßfeld, C. Schwartz, and P. Tran-Gia, “Predicting Result Quality in Crowdsourcing Using Application Layer Monitoring”, in *5th International Conference on Communications and Electronics (ICCE 2014)*, Da Nang, Vietnam, Jul. 2014.

[13] V. Burger, M. Hirth, C. Schwartz, and T. Hoßfeld, “Increasing the Coverage of Vantage Points in Distributed Active Network Measurements by Crowdsourcing”, in *Measurement, Modelling and Evaluation of Computing Systems (MMB 2014)*, Bamberg, Germany, Mar. 2014.

[14] C. Schwartz, M. Scheib, T. Hoßfeld, P. Tran-Gia, and J. M. Gimenez-Guzman, “Trade-Offs for Video-Providers in LTE Networks: Smartphone Energy Consumption Vs Wasted Traffic”, in *22nd International Teletraffic Congress Specialist Seminar on Energy Efficient and Green Networking*, Christchurch, New Zealand, Nov. 2013.

[15] C. Schwartz, F. Lehrieder, F. Wamser, T. Hoßfeld, and P. Tran-Gia, “Smart-Phone Energy Consumption Vs. 3G Signaling Load: The Influence of Application Traffic Patterns”, in *Tyrrhenian International Workshop 2013 on Digital Communications: Green ICT*, Genova, Italy, Sep. 2013.

[16] C. Schwartz, R. Pries, and P. Tran-Gia, “A Queuing Analysis of an Energy-Saving Mechanism in Data Centers”, in *International Conference on Information Networking (ICOIN)*, Bali, Indonesia, Feb. 2012.

[17] D. Hock, M. Hartmann, M. Menth, and C. Schwartz, “Optimizing Unique Shortest Paths for Resilient Routing and Fast Reroute in IP-Based Networks”, in *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, Osaka, Japan, Apr. 2010.

[18] D. Hock, M. Hartmann, C. Schwartz, and M. Menth, “Effectiveness of Link Cost Optimization for IP Rerouting and IP Fast Reroute”, in *15th International GI/ITG Conference on Measurement, Modelling and Evaluation of Computing Systems and Dependability and Fault Tolerance (MMB & DFT 2010)*, Essen, Germany, Mar. 2010.

[19] D. Hock, M. Menth, M. Hartmann, C. Schwartz, and D. Stezenbach, “ResiLyzer: A Tool for Resilience Analysis in Packet-Switched Communication Networks”, in *15th International GI/ITG Conference on Measurement, Modelling and Evaluation of Computing Systems and Dependability and Fault Tolerance (MMB & DFT 2010)*, Essen, Germany, Mar. 2010.

[20] C. Schwartz, J. Eisl, A. Halimi, A. Rafetseder, and K. Tutschku, “Interconnected Content Distribution in LTE Networks”, in *GLOBECOM Workshops, 2010 IEEE*, IEEE, Miami, Florida, USA, 2010, pp. 2028–2033.

[21] M. Hartmann, D. Hock, M. Menth, and C. Schwartz, “Objective Functions for Optimization of Resilient and Non-Resilient IP Routing”, in *7th International Workshop on the Design of Reliable Communication Networks (DRCN)*, Washington, D.C., USA, Oct. 2009.

Software Demonstrations

[22] D. Hock, M. Hartmann, C. Schwartz, and M. Menth, *ResiLyzer: Ein Werkzeug zur Analyse der Ausfallsicherheit in paketvermittelten Kommunikationsnetzen*, Kiel, Germany, Mar. 2011. [Online]. Available: `http://www3.informatik.uni-wuerzburg.de/resilyzer`.

General References

[23] C. Yang, *Huawei Communicate: Weather the Signalling Storm*, White Paper, 2011.

[24] GSM Association and others, *Network Efficiency Task Force Fast Dormancy Best Practices*, White Paper, May 2010.

[25] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "PlanetLab: An Overlay Testbed for Broad-Coverage Services", *Computer Communication Review*, vol. 33, no. 3, 2003.

[26] F. Qian, Z. Wang, A. Gerber, Z. Mao, S. Sen, and O. Spatscheck, "Profiling Resource Usage for Mobile Applications: A Cross-Layer Approach", in *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services (MobiSys)*, Washington, D.C., USA, Jun. 2011.

[27] NokiaSiemensNetworks, *Understanding Smartphone Behavior in the Network*, White Paper, Mar. 2013.

[28] *TS 25.331, Radio Resource Control (RRC); Protocol Specification*, 3GPP, 2012. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/25331.htm.

[29] F. Qian, Z. Wang, A. Gerber, Z. Mao, S. Sen, and O. Spatscheck, "Characterizing Radio Resource Allocation for 3G Networks", in *Proceedings of the 10th Annual Conference on Internet Measurement (IMC)*, Melbourne, Australia, Nov. 2010.

[30] P. Perala, A. Barbuzzi, G. Boggia, and K. Pentikousis, "Theory and Practice of RRC State Transitions in UMTS Networks", in *Proceedings of the Global Communication Conference (GLOBECOM) Workshops*, Honolulu, Hawaii, USA, Nov. 2009.

[31] *TR 22.801, Study on Non-MTC Mobile Data Applications Impacts (Release 11)*, 2011.

[32] F. Qian, Z. Wang, A. Gerber, Z. Mao, S. Sen, and O. Spatscheck, "TOP: Tail Optimization Protocol for Cellular Radio Resource Allocation", in *Proceedings of the 18th IEEE International Conference on Network Protocols (ICNP)*, Kyoto, Japan, Oct. 2010.

[33] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, “Energy Consumption in Mobile Phones: A Measurement Study and Implications for Network Applications”, in *Proceedings of the 9th Annual Conference on Internet Measurement (IMC)*, Chicago, Illinois, USA, Nov. 2009.

[34] C. Lee, J. Yeh, and J. Chen, “Impact of Inactivity Timer on Energy Consumption in WCDMA And CDMA2000”, in *Proceedings of the Wireless Telecommunications Symposium (WTS)*, Pamona, California, USA, May 2004.

[35] T. Tirronen, A. Larmo, J. Sachs, B. Lindoff, and N. Wiberg, “Reducing Energy Consumption of LTE Devices for Machine-to-Machine Communication”, in *Globecom Workshops*, Dec. 2012.

[36] M. Siekkinen, M. Ashraful, J. K. N. Hoque, and M. Aalto, “Streaming over 3G and LTE: How to Save Smartphone Energy in Radio Access Network-Friendly Way”, in *Workshop on Mobile Video*, Feb. 2013.

[37] S. Ickin, K. Wac, M. Fiedler, L. Janowski, J. Hong, and A. Dey, “Factors Influencing Quality of Experience of Commonly Used Mobile Applications”, *IEEE Communications Magazine*, vol. 50, no. 4, pp. 48–56, Apr. 2012.

[38] A. D. Zayas and P. M. Gómez, “A Testbed for Energy Profile Characterization of IP Services in Smartphones over Live Networks”, *Mobile Networks and Applications*, vol. 15, no. 3, pp. 330–343, 2010.

[39] M. Calder and M. Marina, “Batch Scheduling of Recurrent Applications for Energy Savings on Mobile Phones”, in *Proceedings of the Conference on Sensor Mesh and Ad Hoc Communications and Networks (SECON)*, Boston, Massachusetts, USA, Jun. 2010.

[40] E. Vergara and S. Nadjm-Tehrani, “Energy-Aware Cross-Layer Burst Buffering for Wireless Communication”, in *Proceedings of the Conference*

on Future Energy Systems: Where Energy, Computing and Communication Meet, Madrid, Spain, May 2012.

[41] S. Egger, P. Reichl, T. Hoßfeld, and R. Schatz, "Time is Bandwidth? Narrowing the Gap between Subjective Time Perception and Quality of Experience", in *Proceedings of the International Conference on Communications (ICC)*, Ottawa, Canada, Jun. 2012.

[42] S. Egger, T. Hoßfeld, R. Schatz, and M. Fiedler, "Waiting Times in Quality of Experience for Web Based Services", in *Proceedings of the Workshop on Quality of Multimedia Experience (QoMEX)*, Yarra Valley, Australia, Jul. 2012.

[43] P. Tran-Gia, "A Renewal Approximation for the Generalized Switched Poisson Process", in *Proceedings of the International Workshop on Applied Mathematics and Perfomance/Reliability Models of Computer/Communication Systems*, Pisa, Italy, Sep. 1983.

[44] ——, "A Class of Renewal Interrupted Poisson Processes and Applications to Queueing Systems", *Zeitschrift für Operations Research*, vol. 32, no. 3-4, pp. 231–250, 1988.

[45] H. Ascher and H. Feingold, *REPAIRABLE SYSTEMS RELIABILITY: MODELING, INFERENCE, MISCONCEPTIONS AND THEIR CAUSES.* Marcel Dekker, Inc., 1984.

[46] H. Falaki, D. Lymberopoulos, R. Mahajan, S. Kandula, and D. Estrin, "A First Look at Traffic on Smartphones", in *Proceedings of the 10th Annual Conference on Internet Measurement (IMC)*, Melbourne, Australia, Nov. 2010.

[47] S. Spirou and B. Stiller, "Economic Traffic Management", in *Integrated Network Management-Workshops, 2009. IM'09. IFIP/IEEE International Symposium on*, Long Island, New York, USA, Jun. 2009.

[48] Trilogy Project, *D2 - Lessons in 'Designing for Tussle' from Case Studies*, Deliverable of the Trilogy Project ICT-216372, May 2008.

[49] A. Begen, T. Akgul, and M. Baugher, “Watching Video Over the Web: Part 2: Applications, Standardization, and Open Issues”, *Internet Computing*, vol. 15, no. 3, 2011.

[50] I. Elsen, F. Hartung, U. Horn, M. Kampmann, and L. Peters, “Streaming Technology in 3G Mobile Communication Systems”, *IEEE Computer*, vol. 34, no. 9, 2001.

[51] O. Oyman and S. Singh, “Quality of Experience for HTTP Adaptive Streaming Services”, *IEEE Communications Magazine*, vol. 50, no. 4, 2012.

[52] I. Sodagar, “The MPEG-DASH Standard for Multimedia Streaming Over the Internet”, *MultiMedia*, vol. 18, no. 4, 2011.

[53] T. Hoßfeld, R. Schatz, E. Biersack, and L. Plissonneau, “Internet Video Delivery in YouTube: From Traffic Measurements to Quality of Experience”, in *Data Traffic Monitoring and Analysis*, Springer, 2013, pp. 264–301.

[54] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hossfeld, and P. Tran-Gia, “A Survey on Quality of Experience of HTTP Adaptive Streaming”, *IEEE Communications Surveys Tutorials*, vol. PP, Sep. 2014.

[55] C. Sieber, T. Hoßfeld, T. Zinner, P. Tran-Gia, and C. Timmerer, “Implementation and User-centric Comparison of a Novel Adaptation Logic for DASH with SVC”, in *IFIP/IEEE QCMan 2013*, May 2013.

[56] J. Lievens, S. M. Satti, N. Deligiannis, P. Schelkens, and A. Munteanu, “Optimized Segmentation of H. 264/AVC Video for HTTP Adaptive Streaming”, in *IFIP/IEEE QCMan 2013*, Sep. 2013.

[57] T. Hoßfeld, S. Egger, R. Schatz, M. Fiedler, K. Masuch, and C. Lorentzen, “Initial Delay vs. Interruptions: Between the Devil and the Deep Blue Sea”, in *QoMEX 2012*, 2012.

[58] J. Yan, W. Mühlbauer, and B. Plattner, “An Analytical Model for Streaming over TCP”, in *Smart Spaces and Next Generation Wired/Wireless Networking*, Springer, 2011, pp. 370–381.

[59] T. Hoßfeld, F. Liers, R. Schatz, B. Staehle, D. Staehle, T. Volkert, and F. Wamser, “Quality of Experience Management for YouTube: Clouds, FoG and the AquareYoum”, *PIK - Praxis der Informationverarbeitung und - kommunikation*, vol. 35, Aug. 2012.

[60] M. Fiedler, “On the Limited Potential of Buffers to Improve Quality of Experience”, in *6th Int. Workshop on Information Quality and Quality of Service for Pervasive Computing*, Mar. 2014.

[61] T. H. Luan, L. X. Cai, and X. Shen, “Impact of Network Dynamics on User’s Video Quality: Analytical Framework and QoS Provision”, *IEEE Transactions on Multimedia*, vol. 12, no. 1, 2010.

[62] A. E. Essaili, D. Schroeder, D. Staehle, M. Shehada, W. Kellerer, and E. Steinbach, “Quality-of-Experience driven Adaptive HTTP Media Delivery”, in *International Conference on Communications*, Jun. 2013.

[63] X. Li, M. Dong, Z. Ma, and F. C. Fernandes, “GreenTube: Power Optimization for Mobile Videostreaming via Dynamic Cache Management”, in *Conference on Multimedia*, Oct. 2012.

[64] M. Yadin and P. Naor, “Queueing Systems with a Removable Service Station”, *OR*, pp. 393–405, 1963.

[65] Z. G. Zhang and N. Tian, “The N Threshold Policy for the $GI/M/1$ Queue”, *Operations Research Letters*, vol. 32, no. 1, 2004.

[66] K.-H. Wang and J.-C. Ke, “A Recursive Method to the Optimal Control of an $M/G/1$ Queueing System with Finite Capacity and Infinite Capacity”, *Applied Mathematical Modelling*, vol. 24, no. 12, pp. 899–914, 2000.

[67] W. Böhm and S. G. Mohanty, “The Transient Solution of $M/M/1$ Queues Under (M, N) Policy. A Combinatorial Approach”, *Journal of Statistical Planning and Inference*, vol. 34, pp. 23–33, 1993.

[68] T. Hoßfeld, R. Schatz, and S. Egger, “SOS: The MOS is not Enough!”, in *Third International Workshop on Quality of Multimedia Experience (QoMEX 2011)*, IEEE, Mechelen, Belgium, Sep. 2011, pp. 131–136.

[69] F. Wamser, D. Hock, M. Seufert, B. Staehle, R. Pries, and P. Tran-Gia, "Using Buffered Playtime for QoE-Oriented Resource Management of YouTube Video Streaming", *Transactions on Emerging Telecommunications Technologies*, vol. 24, Apr. 2013.

[70] T. Zinner, M. Jarschel, A. Blenk, F. Wamser, and W. Kellerer, "Dynamic Application-aware Resource Management using Software-Defined Networking: Implementation Prospects and challenges", in *IEEE Network Operations and Management Symposium (NOMS)*, May 2014.

[71] M. Zink, K. Suh, Y. Gu, and J. Kurose, "Watch Global, Cache Local: YouTube Network Traffic at a Campus Network: Measurements and Implications", in *Electronic Imaging 2008*, Aug. 2008.

[72] T. Hoßfeld, R. Schatz, E. Biersack, and L. Plissonneau, "Internet Video Delivery in YouTube: From Traffic Measurements to Quality of Experience", in *Data Traffic Monitoring and Analysis: From measurement, classification and anomaly detection to Quality of experience*, Springer, 2012.

[73] T. Hoßfeld, T. Zinner, R. Schatz, M. Seufert, and P. Tran-Gia, "Transport Protocol Influences on YouTube QoE ", University of Würzburg, Tech. Rep. 482, Jul. 2011.

[74] T. Hoßfeld, D. Strohmeier, A. Raake, and R. Schatz, "Pippi Longstocking Calculus for Temporal Stimuli Pattern on YouTube QoE", in *Workshop on Mobile Video*, Feb. 2013.

[75] *TS 36.321, Evolved Universal Terrestrial Radio Access (E-UTRA); Medium Access Control (MAC) Protocol Specification*, 2013.

[76] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, "A Close Examination of Performance and Power Characteristics of 4G LTE Networks", in *Conference on Mobile Systems, Applications, and Services*, 2012.

[77] D. D. Clark, J. Wroclawski, K. R. Sollins, and R. Braden, “Tussle in Cyberspace: Defining Tomorrow’s Internet”, *IEEE/ACM Trans. Netw.*, vol. 13, no. 3, 2005.

[78] M.-N. Garcia, D. Dytko, and A. Raake, “Quality impact due to initial loading, stalling, and video bitrate in progressive download video services”, in *QoMEX 2014*, Sep. 2014.

[79] M. Fiedler, T. Hoßfeld, and P. Tran-Gia, “A Generic Quantitative Relationship Between Quality of Experience and Quality of Service”, *IEEE Network*, vol. 24, no. 2, pp. 36–41, 2010.

[80] S. Möller, *ASSESSMENT AND PREDICTION OF SPEECH QUALITY IN TELECOMMUNICATIONS*. Springer, 2000.

[81] ITU-T P.1201, *Parametric Non-Intrusive Assessment of Audiovisual Media Streaming Quality. Amendment 2: New Appendix III – Use of ITU-T P.1201 for Non-Adaptive, Progressive Download Type Media Streaming*, International Telecommunications Union, Dec. 2013.

[82] H. Qian, D. Medhi, and K. Trivedi, “A Hierarchical Model to Evaluate Quality of Experience of Online Services Hosted by Cloud Computing”, in *Symposium on Integrated Network Management*, 2011.

[83] P. Amrehn, K. Vandenbroucke, T. Hoßfeld, K. De Moor, M. Hirth, R. Schatz, and P. Casas, “Need for Speed? On Quality of Experience for Cloud-based File Storage Services”, 2013.

[84] I. Drago, M. Mellia, M. M. Munafo, A. Sperotto, R. Sadre, and A. Pras, “Inside Dropbox: Understanding Personal Cloud Storage Services”, in *Internet Measurement Conference*, Boston, Massachusetts, USA, Nov. 2012.

[85] J. Wang, H. Zhou, M. Zhou, and L. Li, “A General Model for Long-Tailed Network Traffic Approximation”, *Journal of Supercomputing*, vol. 38, no. 2, 2006.

[86] M. Al-Fares, A. Loukissas, and A. Vahdat, "A Scalable, Commodity Data Center Network Architecture", in *SIGCOMM '08: Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, Seattle, WA, USA, Aug. 2008, pp. 63–74.

[87] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "VL2: A scalable and flexible data center network", *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, pp. 51–62, 2009.

[88] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "BCube: a High Performance, Server-Centric Network Architecture for Modular Data Centers", in *SIGCOMM '09: Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, Barcelona, Spain, Aug. 2009, pp. 63–74.

[89] R. Bolla, R. Bruschi, F. Davoli, and F. Cucchietti, "Energy Efficiency in the Future Internet: A Survey of Existing Approaches and Trends in Energy-aware Fixed Network Infrastructures", *Communications Surveys & Tutorials, IEEE*, vol. 13, no. 2, 2011.

[90] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown, "Elastic tree: Saving energy in data center networks", in *7th USENIX Symposium on Networked System Design and Implementation (NSDI)*, San Jose, CA, USA, Apr. 2010, pp. 249–264.

[91] D. Kliazovich, P. Bounvry, Y. Audzevich, and S. U. Khan, "GreenCloud: A Packet-level Simulator of Energy-aware Cloud Computing Data Centers", in *IEEE Globecom*, Miami, FL, USA, Dec. 2010.

[92] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle, "Managing Energy and Server Resources in Hosting Centers", in *Proceedings of the 18th ACM Symposium on Operating Systems Principles*, Chateau Lake Louise, Banff, Canada, Oct. 2001.

[93] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, and N. Gautam, "Managing Server Energy and Operational Costs in Hosting Centers", *ACM SIGMETRICS Performance Evaluation Review*, vol. 33, no. 1, pp. 303–314, 2005.

[94] M. Mazzucco, D. Dyachuk, and R. Deters, "Maximizing Cloud Providers Revenues via Energy Aware Allocation Policies", in *IEEE International Conference on Cloud Computing*, Miami, Florida, Jul. 2010, pp. 131–138.

[95] D. Dyachuk and M. Mazzucco, "On Allocation Policies for Power and Performance", in *11th ACM/IEEE International Conference on Grid Computing (Grid 2010) – Energy Efficient Grids, Clouds and Clusters Workshop (E2GC2-2010)*, Brussels, Belgium, Oct. 2010.

[96] M. Mazzucco, D. Dyachuk, and M. Dikaiakos, "Profit-Aware Server Allocation for Green Internet Services", in *18th Annual IEEE/ACM International Symposium on Modeling, Analysis, and Simulation of Computer Systems*, Miami, Florida: IEEE Computer Society, Aug. 2010, pp. 277–284.

[97] A. Gandhi, M. Harchol-Balter, and I. Adan, "Server Farms with Setup Costs", *Performance Evaluation*, 2010.

[98] F. Metzger, A. Rafetseder, P. Romirer-Maierhofer, and K. Tutschku, "Exploratory Analysis of a GGSN's PDP Context Signaling Load", *Journal of Computer Networks and Communications*, vol. 2014, Feb. 2014.

[99] M. Shafiq, L. Ji, A. Liu, and J. Wang, "Characterizing and Modeling Internet Traffic Dynamics of Cellular Devices", in *SIGMETRICS*, 2011.

[100] U. Paul, A. Subramanian, M. Buddhikot, and S. Das, "Understanding Traffic Dynamics in Cellular Data Networks", in *Computer Communications*, 2011.

[101] Y. Zhang and A. Årvidsson, "Understanding the Characteristics of Cellular Data Traffic", *SIGCOMM Comput. Commun. Rev.*, vol. 42, Sep. 2012.

[102] P. Svoboda, F. Ricciato, E. Hasenleithner, and R. Pilz, "Composition of GPRS, UMTS Traffic: Snapshots from a Live Network", in *Workshop on Internet Performance, Simulation, Monitoring and Measurement*, 2006.

[103] P. Lee, T. Bu, and T. Woo, "On the Detection of Signaling DoS Attacks on 3G Wireless Networks", in 26th *Conf. on Computer Communications*, 2007.

[104] P. Romirer-Maierhofer, F. Ricciato, and A. Coluccia, "Explorative Analysis of One-way Delays in a Mobile 3G Network", in *Local and Metropolitan Area Networks*, Sep. 2008.

[105] J. Howe, "The Rise of Crowdsourcing", *Wired Magazine*, vol. 14, no. 6, 2006.

[106] T. Hoßfeld, M. Hirth, and P. Tran-Gia, "Modeling of Crowdsourcing Platforms and Granularity of Work Organization in Future Internet", in *International Teletraffic Congress*, 2011.

[107] N. Morrow, N. Mock, A. Papendieck, and N. Kocmich, "Independent Evaluation of the Ushahidi Haiti Project", *Development Information Systems International*, vol. 8, 2011.

[108] M. J. Raddick, G. Bracey, P. L. Gay, C. J. Lintott, P. Murray, K. Schawinski, A. S. Szalay, and J. Vandenberg, "Galaxy Zoo: Exploring the Motivations of Citizen Science Volunteers", *Astronomy Education Review*, vol. 9, no. 1, 2010.

[109] S. Faradani, B. Hartmann, and P. G. Ipeirotis, "What's the Right Price? Pricing Tasks for Finishing on Time", in *Human Computation*, 2011.

[110] J. Wang, S. Faridani, and P. G. Ipeirotis, "Estimating the completion time of crowdsourced tasks using survival analysis models", in *Crowdsourcing for Search and Data Mining*, 2011.

[111] M. S. Bernstein, D. R. Karger, R. C. Miller, and J. Brandt, "Analytic Methods for Optimizing Realtime Crowdsourcing", *ArXiv preprint arXiv:1204.2995*, 2012.

[112] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, “The Cost of a Cloud: Research Problems in Data Center Networks”, *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 89–73, 2009.

[113] United States Environmental Protection Agency, *Data Center Report to Congress*, `http://1.ussa.gov/iEAPHC`, 2007.

[114] P. Tran-Gia and M. Mandjes, “Modeling of Customer Retrial Phenomenon in Cellular Mobile Networks”, *IEEE JSAC special issue on Personal Communication - Services, architecture and performance issues*, vol. pages 1406–1414, 1997.

[115] L. Barroso and U. Holzle, “The Case for Energy-proportional Computing”, *IEEE Computer*, vol. 40, no. 12, pp. 33–37, 2007.

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

[117] F. Ricciato, P. Svoboda, J. Motz, W. Fleischer, M. Sedlak, M. Karner, R. Pilz, P. Romirer-Maierhofer, E. Hasenleithner, W. Jäger, P. Krüger, F. Vacirca, and M. Rupp, “Traffic Monitoring and Analysis in 3G Networks: Lessons Learned from the METAWIN Project”, *E & i Elektrotechnik und Informationstechnik*, vol. 123, no. 7-8, pp. 288–296, 2006.

[118] S. M. and H. Lipson, “Distilling Free-Form Natural Laws from Experimental Data”, *Science*, vol. 324, no. 5923, 2009.

[119] P. D. Ellis, *THE ESSENTIAL GUIDE TO EFFECT SIZES: STATISTICAL POWER, META-ANALYSIS, AND THE INTERPRETATION OF RESEARCH RESULTS*. Cambridge University Press, 2010.

[120] L. Kleinrock, *Queuing Systems*. Wiley, 1975.

[121] D. Manfield and P. Tran-Gia, “Analysis of a Storage System with Batch Input Arising out of Message Packetisation”, *IEEE Transactions on Communications*, vol. 30, 1982.

ISSN 1432-8801