UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

Martin Dräxler

Resource Allocation and Scheduling in Dense Mobile Access Networks

Dissertation

accepted by the

Faculty of Electrical Engineering,
Computer Science, and Mathematics

in partial fulfillment of the requirements for the degree of

Doctor rerum naturalium (Dr. rer. nat.)

Paderborn, April 2015

Referees:
Prof. Dr. Holger Karl, University of Paderborn, Germany
Dr. Vincenzo Mancuso, IMDEA Networks Institute, Madrid, Spain

Additional committee members:
Prof. Dr. Gitta Domik-Kienegger, University of Paderborn, Germany
Prof. Dr. Falko Dressler, University of Paderborn, Germany
Dr. Jens Simon, University of Paderborn, Germany

Submission: 29.04.2015
Examination: 12.06.2015

Abstract

Traffic in wireless access networks has been growing substantially in recent years, both in terms of total volume and of data rate required by individual users. This leads to the deployment of very dense and heterogeneous wireless networks. This growth cannot only be addressed by simply scaling existing networks by orders of magnitude, to fulfill traffic forecasts, due to limited backhaul capacity, increased energy consumption, and explosion of signaling traffic. Thus wireless transmissions have to be coordinated and available wireless resources have to be utilized more efficiently. Furthermore backhaul networks have to be dynamically adapted to the requirements of wireless access networks.

In this thesis, I present two different approaches for the efficient operation of wireless access networks:

First, I introduce *anticipatory download scheduling*, an approach to efficient download scheduling for video segments in high demand video streaming, based on prediction of future available data rates. I investigate this approach with respect to the quality of experience (QoE) for users and also present an extension to use this approach to increase the energy efficiency of mobile access networks.

Second, I introduce *backhaul network reconfiguration* to efficiently allocate backhaul resources based on requirements from wireless coordination. This approach is based on Wavelength-Division-Multiplexed Passive Optical Network (WDM-PON) backhaul networks and increases the feasibility of wireless coordination while reducing the power consumption of the backhaul network.

For both approaches I present a system architecture to integrate the approaches into the existing infrastructure of mobile access networks. Based on these architectures, I also present prototype implementations for both approaches.

Zusammenfassung

Der Datenverkehr in drahtlosen Zugangsnetzen hat sich in den letzten Jahren erheblich vervielfacht, sowohl im Bezug auf die gesamte Verkehrslast, als auch im Bezug auf die geforderte Datenrate pro Benutzer. Dies führt dazu, dass sehr dichte, heterogene drahtlose Zugangsnetze notwendig sind. Dieses Wachstum an Datenverkehr, das auch verschiedene Studien für die Zukunft vorhersagen, kann nicht einfach nur durch ein Verdichten der drahtlosen Zugangspunkte gehandhabt werden, da dadurch mehrere Probleme durch begrenzte Backhaulkapazitäten, erhöhten Energieverbrauch und verstärkt nötige Signalisierung im Netzwerk entstehen. Deshalb müssen zudem die drahtlosen Übertragungen koordiniert werden und die zur Verfügung stehenden drahtlosen Ressourcen effizienter genutzt werden. Weiterhin muss auch das Backhaulnetzwerk dynamisch an die Anforderungen des drahtlosen Zugangsnetzwerks angepasst werden.

In dieser Arbeit stelle ich zwei verschiedene Ansätze vor, um das drahtlose Zugangsnetz effizienter zu betreiben:

Als erstes stelle ich *anticipatory download scheduling* vor, einen Ansatz zur Koordination von effizienten Segment-Downloads für hochauflösendes Video-Streaming, basierend auf der Vorhersage von zukünftig zur Verfügung stehenden Datenraten. Ich untersuche diesen Ansatz in Bezug auf die Dienstgüte für die Nutzer und stelle zudem eine Erweiterung vor, um die Energieeffizienz des drahtlosen Zugangsnetzes zu erhöhen.

Als zweites stelle ich *backhaul network reconfiguration* vor, um die Ressourcen des Backhaulnetzes effizient, auf Basis der Anforderungen des drahtlosen Zugangsnetzes, zuzuteilen. Dieser Ansatz basiert auf Wavelength-Division-Multiplexed Passive Optical Network (WDM-PON) Backhaul-Netzen und erhöht die Realisierbarkeit von drahtloser Koordination. Zudem verringert dieser Ansatz den Energieverbrauch des Backhaulnetzwerks.

Für beide Ansätze stelle ich konkrete Systemarchitekturen vor, um die Ansätze in die bestehende Zugangsnetz-Infrastruktur zu integrieren. Auf Basis dieser Systemarchitekturen habe ich zudem einen Prototypen für jeden der Ansätze implementiert.

Acknowledgements

First of all, I would like to thank Holger Karl for supervising me during my work that led to this thesis. He was always open for interesting discussions, always encouraged me with my work, and was always available when I needed advice.

I would also like to thank my current and former colleagues in the Research Group Computer Networks at the University of Paderborn, in the EU project CROWD and the SPAN collaboration for all the insightful discussions and shared experiences. Especially I would like to thank Vincenzo Mancuso for being available as the second examiner of this thesis, and Thorsten Biermann and Stefan Valentin for the inspiring discussions that eventually led to the approaches and results in this thesis.

Finally, I am very thankful for Sevil always being there for me, always loving me and always supporting me with my work and my thesis. Also, I would have never finished my PhD without the support from my family and all the friends in different parts of the world.

Contents

List of Figures

List of Tables

List of Algorithms

1

Introduction

1.1 Current Trends and Issues in Mobile Access Networks

Recent studies [Cis12, Cis14a, Cis14b, Law13] on the development of mobile access networks indicate a steep increase in volume of traffic that has to be processed and transmitted by network equipment. This is due to two trends: 1. increasing data rate demands from users and consequently 2. a densification of mobile access networks. The approaches presented in this thesis deal with the issues arising from these two trends. Both trends are described in more detail in this section.

1.1.1 Increasing Data Rate Demands

The most recent "Global Mobile Data Traffic Forecast" [Cis14b] published by Cisco in 2014 provides very interesting insights into current and future developments of traffic in mobile access networks. In 2013, global mobile data traffic increased by 81%, from 820 petabytes per month in 2012 to 1,5 exabytes per month in 2013, which corresponds to 18 times as much as the monthly traffic of

the entire Internet in 2000. More than half (53%) of the mobile data traffic in 2013 was video traffic.

For the year 2018, the study predicts a monthly global mobile data traffic of more than 15 exabytes and a 69% share of video traffic over mobile networks.

The infrastructure of mobile access networks has to keep up with this trend and provide the data rates to satisfy the increasing demands. This is not feasible by only using more and more network equipment or by simply increasing the physical layer capacity of wireless transmissions (e.g, using more spectrum). In addition, coordination and scheduling approaches on the application layer have to be used to utilize the physical resources of the wireless channel more efficiently [ASM15, SPJFL15]. As video traffic already represents half of the traffic in 2013 and is expected to increase to over two thirds in 2018, the first part of this thesis is focused especially on mobile video traffic.

In addition to the limited wireless channel resources, the backhaul network, which transports the data between the radio access network and the core network, has to be considered. With the increase in traffic handled by the radio access network, the traffic load on the backhaul network also increases significantly. This traffic load on the backhaul network is even increased further by the control information that has to be exchanged between the Base Stations (BSs) to coordinate and schedule the wireless transmissions and the backhaul network will become a bottleneck for mobile traffic [TMF+14, JMJ+13, BSC+11, BSC+12]. Thus, in the second part, this thesis addresses the efficient allocation of backhaul network resources.

1.1.2 Densification of Mobile Access Networks

In conjunction with increasing data rate demands, there is also an increase in the number of network devices, referred to as *densification* of mobile access networks. According to a recent study by Cisco [Cis14b] there are globally 7 billion User Equipments (UEs) (smartphones, tablets, etc.) with an average downlink data rate of 1.4 Mb/s per UE. 30% of traffic is transmitted over 4G networks and 45% of traffic is offloaded to WiFi access points or femto cells. The study predicts for 2018 that the number of UEs will increase to 10 billion with an average downlink data rate of 2.5 Mb/s per UE. 51% of traffic will be transmitted over 4G networks and over 50% is offloaded to WiFi access points or femto cells.

This *densification* of UEs cannot just be met by new generations of physical layer technologies, but has to be met by a *densification* of mobile access networks [And13, BLM+14]. This means that more base stations have to be deployed with heterogeneous cell sizes (macro, pico, femto) and heterogeneous radio technologies (4G, 5G, WiFi). Such a densified deployment of base stations together with the increase in mobile traffic leads to two major issues: increased potential for harmful interferences and a backhaul bottleneck [TMF+14, JMJ+13, BSC+11, BSC+12].

Harmful interference cannot be avoided by simple, static coordination techniques like frequency reuse, because usually full frequency reuse is required to achieve the desired data rates inside one cell. Instead, radio transmissions have to be coordinated dynamically. Wireless coordination schemes require high data

rate and low-latency access to base stations to exchange coordination information. This coordination information is exchanged over the backhaul network together with the normal UE traffic. With the predicted increase in UE traffic, the backhaul will likely become a bottleneck if the use of backhaul resources is not also coordinated in a dynamic way.

1.2 Scope & Goals

The issues introduced in the previous section need to be addressed by different approaches in different areas of the mobile access network. Both the approaches and the parts of the mobile access network are related to each other, as illustrated in Figure 1.1:

First of all there is the tight relation between the radio access network (on the left) and the backhaul network (on the right) because the traffic from the radio access network has to be handled by the backhaul network. Approaches for control mechanisms (indicated by gears) for both the radio access network and the backhaul network have to be implemented in the backhaul network because for both domains the control information has to be exchanged via the backhaul network. In this thesis I do not develop new control mechanisms for the radio access network and focus on control mechanisms for the backhaul network to optimize both the utilization of backhaul network resources and to facilitate the implementation of control mechanisms for the radio access network.

Furthermore, the applications running on the UEs (indicated by filmstrips) largely determine the amount of traffic that has to be handled by the radio access network and consequently the backhaul network as well. Thus in this thesis I also focus on approaches to make the applications "smarter" with respect to the amount of resources of the mobile access network they use.

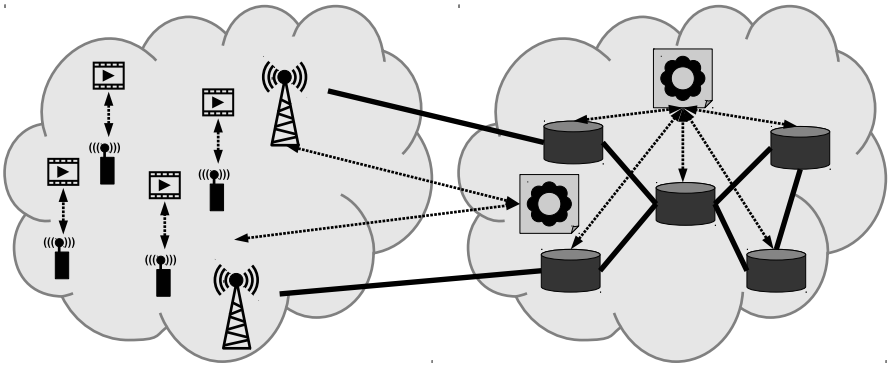


Figure 1.1: Scope of the thesis

For the new approaches that I present in this thesis I define the following overall goals with respect to the previously introduced scope and issues:

1. **Efficient use of wireless resources**
   Applications should use the available wireless resources as efficiently as pos-

sible. Efficiency implies that no wireless resources are allocated which could be used elsewhere or are allocated but not used by an application, e.g., data is transmitted over the wireless channel but later discarded by the application.

2. **Efficient allocation of backhaul resources**
   Backhaul resources should be allocated in a way that no resources are allocated and not utilized while they could be allocated and utilized elsewhere.

3. **Limited QoE degradation**
   The quality of experience (QoE) for users should not be degraded as the result from using a new approach presented in this thesis. Degraded QoE means, for example, reducing the visual quality of a streamed video or frequent interruptions of the playback.

4. **Reduced energy consumption**
   As long as there are unused resources, network equipment should be powered off to reduce the energy consumption of the network, both for the mobile access network and the backhaul network.

In the final chapter of my thesis, after describing my approaches and presenting the evaluation results, I discuss how my approaches help in achieving these goals.

1.3 Approaches

My approach for the application area is *anticipatory download scheduling*. For the problem area of the backhaul network, I focus on *backhaul reconfiguration*. I introduce both approaches in this section.

1.3.1 Anticipatory Download Scheduling

Radio access networks usually employ the prediction of future states of the wireless channel into the scheduling of wireless channel resources. This channel prediction is bound to a small horizon of few milliseconds and has to be very precise to allow fine-grain scheduling of wireless channel resources. This idea of channel prediction can be extended to the prediction of future available data rates to larger horizons of multiple seconds or even minutes. Such a data rate prediction does not have to be as precise as the channel prediction. Instead of using the prediction to schedule channel resources, data rate prediction can be used to make scheduling decisions for applications running on the UEs. Because these scheduling decisions employ knowledge of future available data rates, I call them *anticipatory*.

Such anticipatory decisions are especially possible with applications which continuously download data and where the data that has to be downloaded in the future is known in advance. The most straightforward example are streaming applications for video or audio, but also more complex applications like on-demand games are possible.

My approach is built around the application of mobile video streaming, in particular, segmented video streaming. In segmented video streaming, a video player application downloads the video file not as a continuous stream of data, but divided into multiple segments of a fixed duration. Normally, these segments are downloaded constantly during the playout. A small amount of segments is usually buffered by the player application to account for short phases of small available data rate. The video provider can also make segments available in multiple video quality levels, which result in different file sizes for the segments. Usually the video player application selects the quality of each segment, based on a *retrospective* analysis of the available data rate.

This simple buffering scheme together with the *retrospective* selection of video quality levels is not sufficient to deliver an acceptable QoE to the users and to use available wireless resources efficiently, because it does not take any variation of future available data rates into account. Thus I extend segmented video streaming by an *anticipatory* approach to schedule both the download time and the video quality of each segment, based on the prediction of future available data rates. Future available data rates refers to the available data rates for a user in the next seconds or minutes, not to data rates available through new technologies in the next years.

I describe and evaluate my approach in Chapters 4 to 6 and present a prototype implementation in Chapter 7.

1.3.2 Backhaul Reconfiguration

Wireless resources can only be efficiently utilized if there is a backhaul network that provides sufficient capacity and sufficiently low latency to both transport the user data to the BSs and to exchange coordination information between coordinated BSs. I refer to this set of coordinated BSs as CBS. In order to achieve these capabilities in the backhaul network, the idea of *backhaul reconfiguration* has been introduced [DBK13]. With backhaul reconfiguration, resources in the backhaul network are assigned proportional to actual demands.

I have extended the basic idea of *backhaul reconfiguration* in three novel ways: first, I explicitly model the assignment of optical wavelengths in WDM-PON backhaul networks (Chapter 9); second, I add the capability to manage the backhaul for *dense* wireless access networks with hotspots of users (Chapter 10); third, I show how backhaul reconfiguration can be implemented together with Software-Defined Network (SDN) based on a prototype implementation (Chapter 11).

1.4 Contributions

My contributions in this thesis are divided into three areas. I have published or co-authored five conference papers related to *anticipatory download scheduling* and four conference papers related to *backhaul reconfiguration*. Additionally, I have co-authored five conference papers and two journal papers on background and related work.

Anticipatory Download Scheduling

- M. Dräxler, J. Blobel, and H. Karl. Anticipatory Download Scheduling in Wireless Video Streaming with Uncertain Data Rate Prediction. In *Proceedings of the 8th IFIP Wireless and Mobile Networking Conference (WMNC)*, 2015. submitted.
- M. Dräxler, J. Blobel, P. Dreimann, S. Valentin, and H. Karl. SmarterPhones: Anticipatory Download Scheduling for Wireless Video Streaming. In *Proceedings of the International Conference on Networked Systems (NetSys)*, 2015.
- M. Dräxler, P. Dreimann, and H. Karl. Anticipatory Power Cycling of Mobile Network Equipment for High Demand Multimedia Traffic. In *IEEE Online Conference on Green Communications (IEEE Online GreenComm'14)*, 2014.
- M. Dräxler and H. Karl. SmarterPhones: Anticipatory Download Scheduling for Segmented Wireless Video Streaming. In *1st KuVS Workshop on Anticipatory Networks*, 2014.
- M. Dräxler, J. Blobel, P. Dreimann, S. Valentin, and H. Karl. Anticipatory Buffer Control and Quality Selection for Wireless Video Streaming. *arXiv preprint arXiv:1309.5491v2*, 2014.
- M. Dräxler and H. Karl. Cross-Layer Scheduling for Multi-Quality Video Streaming in Cellular Wireless Networks. In *Proceedings of the 9th IEEE International Wireless Communications and Mobile Computing Conference (IWCMC)*, 2013.

Backhaul Reconfiguration

- S. Auroux, M. Dräxler, A. Morelli, and V. Mancuso. Dynamic network reconfiguration in wireless DenseNets with the CROWD SDN architecture. In *Proceedings of the 2015 European Conference on Networks and Communications (EuCNC)*, 2015.
- M. Dräxler and H. Karl. Dynamic Backhaul Network Configuration in SDN-based Cloud RANs. *arXiv preprint arXiv:1503.03309*, 2015.
- M. I. Sanchez, A. Asadi, M. Dräxler, R. Gupta, V. Mancuso, A. Morelli, A. de la Oliva, and V. Sciancalepore. Tackling the increased density of 5G networks; the CROWD approach. In *IEEE 81st Vehicular Technology Conference: VTC2015-Spring, First International Workshop on 5G Architecture (5GArch 2015)*, 2015.
- M. Dräxler and H. Karl. Feasibility of Base Station Coordination and Dynamic Backhaul Network Configuration in Backhaul Networks with Limited Capacity. In *European Wireless 2014 (EW2014)*, 2014.

- P. Wette, M. Dräxler, A. Schwabe, F. Wallaschek, M. Hassan Zahraee, and H. Karl. MaxiNet: Distributed Emulation of Software-Defined Networks. In *Proceedings of the 2014 IFIP Networking Conference (Networking 2014)*, 2014.

Background and Related Work

- F. Beister, M. Dräxler, J. Aelken, and H. Karl. Power model design for ICT systems – A generic approach. *Computer Communications*, 50:77–85, September 2014.

- C. Cicconetti, A. Morelli, M. Dräxler, H. Karl, V. Mancuso, V. Sciancalepore, R. Gupta, A. de la Oliva, I. Sánchez, P. Serrano, and L. Roullet. The Playground of Wireless Dense Networks of the Future. In *Future Network and MobileSummit 2013*, 2013.

- M. Dräxler, T. Biermann, and H. Karl. Improving Cooperative Transmission Feasibility by Network Reconfiguration in Limited Backhaul Networks. *International Journal of Wireless Information Networks*, 20(3):183–194, 2013.

- H. Ali-Ahmad, C. Cicconetti, A. de la Oliva, M. Dräxler, R. Gupta, V. Mancuso, L. Roullet, and V. Sciancalepore. CROWD: An SDN Approach for DenseNets. In *Second European Workshop on Software Defined Networks*, 2013.

- A. de la Oliva, A. Morelli, V. Mancuso, M. Dräxler, T. Hentschel, T. Melia, P. Seite, and C. Cicconetti. Denser networks for the Future Internet, the CROWD approach. In *MONAMI OConS Workshop: Workshop on Open Connectivity Services for the Future Internet*, 2012.

- M. Dräxler, T. Biermann, H. Karl, and W. Kellerer. Cooperating Base Station Set Selection and Network Reconfiguration in Limited Backhaul Networks. In *Proceedings of the IEEE 23nd International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2012.

- M. Dräxler, F. Beister, S. Kruska, J. Aelken, and H. Karl. Using OMNeT++ for Energy Optimization Simulations in Mobile Core Networks. In *Proceedings of the 5th International ICST Conference on Simulation Tools and Techniques (SIMUTOOLS)*, 2012.

1.5 Structure of the Thesis

The remainder of this thesis is structured as follows:

- **Chapter 2: Technical Background**
  In this chapter, I explain the technical background required for the following chapters. I focus on application layer techniques, wireless access networks and backhaul network technologies.

- **Chapter 3: State of the Art & Related Work on Anticipatory Download Scheduling**
  This chapter contains an overview of related work and other state of the art approaches for anticipatory download scheduling.

- **Chapter 4: Anticipatory Download Scheduling with Perfect Prediction**
  In this chapter, I introduce anticipatory download scheduling with perfect prediction of future available data rates. I present an optimal solution based on a mixed integer quadratically constrained program (MIQCP) and a heuristic solution called Fill algorithm. Both solutions are evaluated and compared via simulation.

- **Chapter 5: Anticipatory Download Scheduling with Uncertain Prediction**
  In contrast to the previous chapter, I focus on anticipatory download scheduling with uncertain prediction. I present a dynamic algorithm called Plan. The algorithm is evaluated in a simulation and compared to the solutions with a perfect prediction.

- **Chapter 6: Anticipatory Download Scheduling for Energy Efficiency**
  Anticipatory download scheduling can be used together with power cycling of base stations to reduce the energy consumption of mobile access networks. In this chapter, I present an extension to the MIQCP from Chapter 4 to incorporate a base station power cycling scheme. Additionally, I present a 2-phase heuristic algorithm. Both solutions are evaluated via simulation.

- **Chapter 7: Anticipatory Download Scheduling Prototype**
  In this chapter I, describe how anticipatory download scheduling can be implemented in a real mobile access network and present a backwards-compatible extension to the HTTP Live Streaming (HLS) protocol. Finally, I show how anticipatory download scheduling with perfect prediction works in a small testbed prototype.

- **Chapter 8: State of the Art & Related Work on Backhaul Network Reconfiguration**
  This chapter contains an overview of related work and other state of the art approaches for backhaul network reconfiguration.

- **Chapter 9: Backhaul Network Reconfiguration**
  In this chapter, I introduce backhaul network reconfiguration and show how it can be implemented both as an integer linear problem (ILP) and a heuristic algorithm, called the BFS algorithm. Both implementations are evaluated and compared in a simulation, with which I also investigate the energy efficiency of the approach.

- **Chapter 10: Backhaul Network Reconfiguration Extension for DenseNets**
  To enable the backhaul network reconfiguration heuristic to work in dense networks with hotspots of users, I have extended the heuristic with a prioritization mechanism, which I present and evaluate in this chapter.

- **Chapter 11: Backhaul Network Reconfiguration Prototype**
  In this chapter I show how backhaul reconfiguration can be implemented on top of an emulated OpenFlow backhaul network.

- **Chapter 12: Conclusion & Future Research Directions**
  In this final chapter, I review how both previously presented approaches, anticipatory download scheduling and backhaul reconfiguration contribute to achieving the goals of this thesis. Then I conclude the work presented in previous chapters and give an outlook on further research directions.

2

Technical Background

In this chapter, I give an overview of the technical background for my approaches which I present in the following chapters. Here, I focus solely on the technical background. I give an overview of related work and state of the art for the anticipatory download scheduling and the backhaul reconfiguration separately in Chapters 3 and 8 respectively.

The technical background in this chapter is split into four sections: backhaul networks, wireless access networks, wireless coordination, and applications.

2.1 Backhaul Networks

The backhaul network describes the part of the mobile access network between the radio access network and the core network. The backhaul network mainly has two

tasks: routing all user data traffic between the base stations and the core network and exchanging control information between the Base Stations (BSs), including control information required for wireless coordination.

A backhaul network can be implemented using wireless technologies like Free Space Optics (FSO) or Point-to-Point (PtP) microwave or wired technologies (optical fiber or cables). Because of the high capacity demands for the backhaul network, especially in future wireless access networks with wireless coordination, optical technologies are the most promising candidate technology. The GreenTouch project [Gre13a] explicitly names Passive Optical Networks (PONs) as the backhaul technology to meet future capacity demands and to provide means for energy savings. In this thesis I focus on Wavelength-Division-Multiplexed Passive Optical Networks (WDM-PONs) as the backhaul technology.

2.1.1 WDM-PON

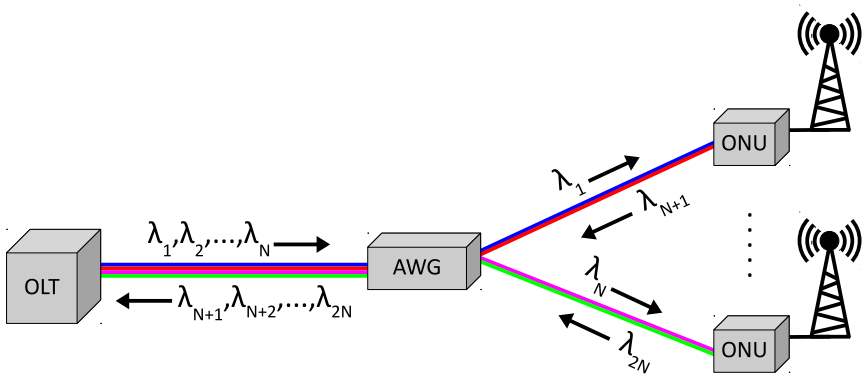


Figure 2.1: WDM-PON example

WDM-PON [RSS10] is an optical fiber technology that multiplexes optical carrier signals on an optical fiber using different wavelengths or colors of laser light.

In a WDM-PON there is one Optical Line Terminal (OLT) which modulates the data on an optical fiber to be received by multiple Optical Network Units (ONUs). To split an optical signal for all ONUs the WDM-PON uses an Array Waveguide Grating (AWG). An AWG is capable of routing individual wavelengths to the ONUs. Thus the OLT modulates the data on the fiber using different wavelengths for the ONUs. Different technological implementations how the wavelengths are split are described by Banerjee et al. [BPC+05]. Figure 2.1 shows an example where each ONU receives data from the OLT using one wavelength λ_i and sends data back using a different wavelength λ_{N+i}. Based on this scheme WDM-PON provides individual point-to-point connections between the OLT and the ONUs, which use their own wavelength. Thus one wavelength does not have to be shared among multiple ONUs. This allows great flexibility in deploying and operating backhaul networks. Wavelengths can be added, dropped or ma-

nipulated in network nodes, enabling to build dynamic topologies in backhaul networks. This flexibility is mainly enabled by the fact that Wavelength Division Multiplexing (WDM) allows operations like multiplexing different wavelengths and converting wavelengths in a purely optical way, avoiding slow and energy-consuming optical-electrical-optical conversions. WDM-PON networks provide a splitting ratio of 40 ONUs per OLT and typically achieve transmission rates up to 2.5 Gb/s per wavelength [BPC+05], but also 10 Gb/s are theoretically possible [GE08].

2.1.2 Power Consumption

The flexibility of WDM-PON makes it possible to model the power consumption of backhaul networks in a fine grain way. Individual Transmission and Reception Unit (TRX) ports of an OLT can be disabled if the wavelength of that port is not assigned to an ONU. Also, a whole ONU can be disabled if the connected BS is powered off. Table 2.1 gives an overview of the power consumption of WDM-PON components.

Table 2.1: WDM-PON components power consumption [GRA+11]

Component	Power consumption
Complete OLT with 40 wavelengths at 1 Gb/s	20 W
Individual OLT TRX port	0.5 W
AWG	-
ONU TRX with 1 Gb/s tunable laser	1 W

Wang et al. [WLLM13] showed that reducing the power consumption is possible by reducing the number of used wavelengths and switching off unused components in the WDM-PON. The results indicate a potential energy saving of 40 % in the investigated scenario.

2.1.3 Software Defined Networks

Software-Defined Networks (SDNs) are based on the concept of decoupling network control and packet forwarding in a network, i.e., decoupling of making forwarding decisions and executing the actual forwarding. The actual forwarding is executed in the network elements, i.e., switches, and forwarding decisions are made by an SDN controller. Consequently, the network control can be directly programmed and the physical network infrastructure is abstracted for applications and network services.

The Open Networking Foundation (ONF) describes the different components of the SDN architecture [ONF12, ONF13], depicted in Figure 2.2, as follows:

- The *SDN Controller* is the central entity responsible for translating the requirements from *SDN Applications* down to network elements and additionally providing *SDN Applications* with an abstract view of the network.

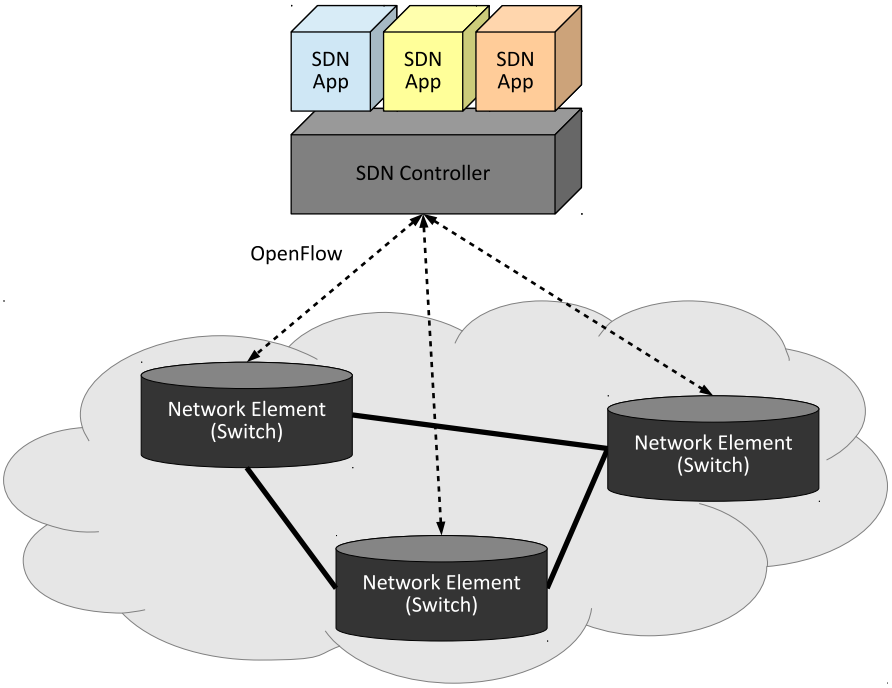


Figure 2.2: SDN architecture [ONF13]

- *SDN Applications* (Apps) are software programs that explicitly, directly, and programmatically communicate their network requirements and their desired network behavior to the *SDN Controller*. They may build an abstracted view of the network for their internal decision making process.
- An SDN-enabled *network element* (e.g., a switch) exposes visibility and control over its forwarding and data processing capabilities. The exposed data can include all or a subset of the physical resources of the network element.

The OpenFlow protocol [MAB+08] is the best-known protocol to implement the communication between the SDN controller and network elements. A number of platforms for the SDN controller have been developed, including OpenDayLight (ODL) [MTVG14], which I use for my prototype implementation in this thesis.

Version 1.4 of the OpenFlow specification [OF 13] includes interfaces to expose the capabilities of a WDM-PON switch, including control of optical switch ports, their wavelength assignment, and power management.

2.2 Wireless Access Networks

Due to the fact that none of the approaches in this thesis modifies existing protocols or standards for the wireless communication in wireless access networks, I introduce two existing, basic models for the wireless channel. For the backhaul reconfiguration for base station coordination, I am only interested in the Signal

to Interference and Noise Ratio (SINR) between a User Equipment (UE) and its surrounding BSs, in order to determine which BSs should cooperate for serving a particular UE. For the anticipatory download scheduling I am furthermore interested in the data rate at which a UE can download from a given BS.

A radio model for wireless access networks includes the following aspects, as illustrated in Figure 2.3:

- *Path loss*: attenuation of the transmitted signal due to the distribution of the signal.
- *Shadowing*: attenuation of the transmitted signal due to obstacles
- *Noise*: general modification of the transmitted signal.
- *Interference*: modification of the transmitted signal due to other transmissions.

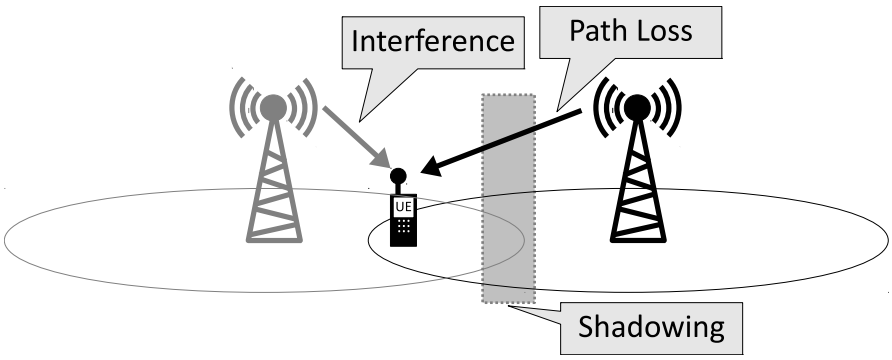


Figure 2.3: Radio model components

2.2.1 Simple 3GPP Radio Model

The 3rd Generation Partnership Project (3GPP) has published a simple Long Term Evolution (LTE) wireless radio model [3GP09]. It calculates the path loss in dB L between the a UE and a BS as

$$L = 128.1 + 37.6 \cdot \log_{10}(d) + S_{\text{ln}} \tag{2.1}$$

where d represents the distance between the UE and the BS's antenna in kilometers. S_{ln} is a normal random variable with zero mean and standard deviation of 10 dB to model slow fading as part of the path loss. Given a known transmission power P in dBm for each BS, it is possible to calculate the received signal strength for all BSs in the range of a UE as $P - L$.

By using Shannon's equation, it is also possible to calculate an achievable data rate for each BS as

$$D = B \cdot \log_2 \left(1 + \frac{P - L}{B \cdot (N + I)}\right) \tag{2.2}$$

where B is the allocated channel bandwidth for the UE, N the noise power per Hertz, and I the simplified average interference per Hertz.

2.2.2 GreenTouch Radio Model

The GreenTouch project has also compiled a wireless radio model [Gre13b] that especially incorporates power consumption and is useful for evaluations related to energy efficiency.

Similar to the 3GPP radio model, the received signal power at a UE u from a BS's antenna a is calculated as

$$rx\_power_{u,a} = tx\_power_a - coupling\_loss_{u,a} \tag{2.3}$$

where tx_power_a is the transmission power of the BS, depending on its type, i.e., macro or pico. *coupling_loss* is the amount of signal power loss that occurs between the time a signal is generated at the BS and the time it is received at the UE. It is defined as

$$\begin{aligned} coupling\_loss_{u,a} = &- max\_gain_a + wall\_penetration \\ &+ path\_loss_{u,a} + shadowing_{u,a} \\ &- directed_{u,a} \end{aligned} \tag{2.4}$$

where max_gain_a is the gain in signal strength produced by the radio frequency amplifier after the signal generation. *wall_penetration* is a signal attenuation depending on frequency and the UE's position in a building. The parameter $directed_{u,a}$ attenuates the signal further if the UE is outside of the directed antenna's angle. The values are calculated using a 3D antenna model as described by the 3GPP [3GP09]. The values for these parameters can be found in the GreenTouch documents [Gre13b] and are omitted here for confidentiality reasons.

Path loss between the UE and the BS is calculated as

$$path\_loss_{u,a} = path\_loss\_constant_a + path\_loss\_factor_a \cdot \log_{10} d \tag{2.5}$$

where $path\_loss\_constant_a$ and $path\_loss\_factor_a$ are frequency-dependent parameters from the GreenTouch documents [Gre13b] and d is the distance between the UE and the BS in kilometers.

Additionally, the model includes shadowing, which is the random attenuation of the signal due to obstacles. It is computed by using two normally distributed random variables. The first random variable represents shadowing due to the UE's position and is the same for all BSs. The second random variable represents shadowing for the connection between the UE and the BS and is calculated for each BS.

$$shadowing_{u,a} = \frac{\left(\overbrace{\mathcal{N}\left(\mu = 0, \sigma = 10\right)}^{\textbf{UE position}} + \overbrace{\mathcal{N}\left(\mu = 0, \sigma = 10\right)}^{\textbf{UE BS connection}} \right)}{\sqrt{2}} \tag{2.6}$$

Both random variables use the same parameters $\mu = 0$ and $\sigma = 10$.

In addition to path loss and shadowing, the GreenTouch model also includes interference as the sum of received power from all antennas A, except the currently connected antenna a.

$$interference_{u,a} = \text{W}\mapsto\text{dBm}\left(\sum_{x \in A \setminus b} \text{dBm}\mapsto\text{W}\left(rx\_power_{u,x}\right)\right) \tag{2.7}$$

To sum up the individual $rx\_power_{u,x}$ values, they have to be converted from dBm to Watt and vice versa using dBm↦W() and W↦dBm() respectively.

The GreenTouch model also includes noise as

$$noise = noise\_per\_hz + \text{W}\mapsto\text{dBm}(bandwidth) + rx\_noise\_figure \tag{2.8}$$

where *noise_per_hz* and *bandwidth* are again taken from the parameters in the GreenTouch documents [Gre13b]. The parameter *rx_noise_figure* models signal attenuation due to the signal receiving process of user equipments.

Finally, the SINR can be calculated as

$$\begin{aligned} sinr_{u,a} =& rx\_power_{u,a} \\ &- \text{W}\mapsto\text{dBm}\left(\text{dBm}\mapsto\text{W}\left(interference_{u,a}\right) + \text{dBm}\mapsto\text{W}\left(noise\right)\right) \end{aligned} \tag{2.9}$$

In order to obtain a data rate from the SINR value, the GreenTouch model uses a lookup table to convert the SINR of each UE to a spectral efficiency value in bits per second per Hertz (Bits/s/Hz). The model includes different lookup tables for different antenna configurations (2x2, 4x2 and 8x2 MIMO) [Gre13b], which are omitted here for confidentiality reasons. Together with a value for the allocated bandwidth per UE from the wireless resource scheduler, the achievable data rate can be calculated from the spectral efficiency.

2.2.3 Power Consumption

The GreenTouch model defines two power consumption models for BSs. One is based on data from the year 2010 and the other one is a forecast of energy consumption of BSs in 2020 [Gre13b, Gre13a]. The 2010 model only considers macro BSs while the 2020 model also considers pico BSs.

Both models include three different states for a BS:

- *Sleep* where the BS is turned off.
- *No Load* where the BS is turned on but completely idle.
- *Full Load* where the BS is fully utilized.

For any other load situation between *No Load* and *Full Load* the power consumption is linearly interpolated. The values for the load states are listed in Table 2.2.

I use the model from the GreenTouch document [Gre13b] with the values in Table 2.2 for my simulations in this thesis. Recently, the GreenTouch project has internally released a document [Gre14] with updates to the model parameters which result in different values for the power consumption. I use the older version for all my simulations to maintain comparability of the results.

Table 2.2: Green Touch power consumption model[Gre13b, Gre13a]

BS type	Sleep	No Load	Full Load
Macro BS 2010	648 W	712 W	1394 W
Macro BS 2020	157 W	189 W	665 W
Pico BS 2020	2 W	4 W	11 W

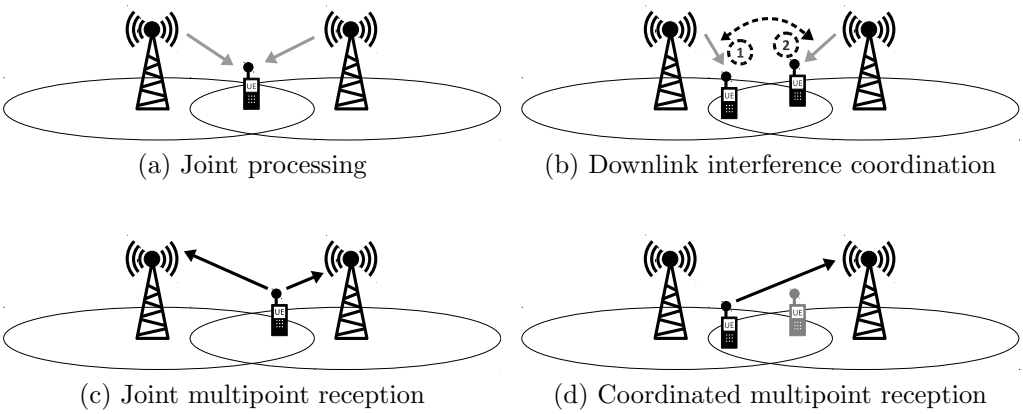
2.3 Wireless Coordination

Wireless coordination is a broad term to describe the paradigm shift in mobile access networks, from coordination of transmission and reception of a single BS to joint coordination of transmissions and receptions of multiple BSs. A large number of different approaches for wireless coordination have been investigated and presented. I do not present any new approaches in the wireless domain and focus on the impact of wireless coordination in the backhaul network. Therefore, I only give a broad overview of two approaches for wireless coordination here: *Coordinated MultiPoint transmission and reception (CoMP)* and *Software Defined Base Station Coordination*.

2.3.1 Coordinated Multipoint Transmission and Reception

CoMP refers to a number of different coordination schemes, which I describe individually in this section. A detailed description how to integrate CoMP into mobile access networks is given by Sawahashi et al. [SKM+10] and Deb et al. [DMMS14]. CoMP schemes can be implemented for both downlink (joint processing and downlink interference coordination) and uplink (joint multipoint reception and coordinated multipoint reception) connections as follows.

- *Joint Processing*
  If two or more BSs are coordinated to perform joint transmission, as shown in Figure 2.4a, all participating BSs send downlink data to the UE synchronously. Hence the UE receives the downlink signal with a higher signal power and the transmission is less prone to harmful interference. Joint transmission requires that the downstream data is present at all participating BSs, resulting in an increased demand for backhaul capacity. Additionally, tight coordination of the participating BSs, including the exchange of Channel State Information (CSI), requires low latency in the backhaul network.



(a) Joint processing (b) Downlink interference coordination

(c) Joint multipoint reception (d) Coordinated multipoint reception

Figure 2.4: Coordinated Multipoint Transmission and Reception

- *Downlink Interference Coordination*
  When multiple UEs are served by multiple BSs, different signals from the BSs cause harmful interference that decreases the wireless throughput. The BSs need to coordinate their transmissions in order to reduce the interference. As shown in Figure 2.4b, BSs can, for example, schedule their transmission to different time slices. The use of Physical Resource Blocks (PRBs) can also be coordinated between the BSs and a technique called coordinated beamforming can reduce interference on the physical level of the transmitted waveform. Downlink interference coordination also requires the exchange of additional information via the backhaul network.

- *Joint Multipoint Reception*
  For joint multipoint reception multiple BSs combine the data they have received from individual UEs. Additionally, if multiple UEs cause interference, as shown in Figure 2.4c, the BSs are able to filter out interference by using interference rejection combining. Similar to joint transmission, this time in the reverse direction, all received data has to be sent from multiple BSs over the backhaul resulting in more required backhaul capacity.

- *Coordinated Multipoint Reception*
  Analogously to coordinated transmissions for BSs, it is also possible to coordinate the uplink transmissions from multiple UEs. For example, transmissions from the UEs can be scheduled to only occur from a single UE at a time, as shown in Figure 2.4d. In that case, the coordination information also has to be exchanged by the base stations over the backhaul network.

Overall, schemes with joint transmission and reception provide a higher gain in terms of efficient utilization of physical resources compared to simple coordination schemes, but also require significantly more capacity in the backhaul network. This makes these schemes especially interesting for the approaches I present in this thesis.

2.3.2 Software Defined Base Station Coordination

The idea behind CoMP has also been developed further to allow new coordination schemes between BSs. There is especially the idea to apply the SDN paradigm to the coordination of BSs. The control of coordination mechanisms is moved to a controller node, thus backhaul capacity constraints similar to CoMP also apply between the controller node and the BSs. A number of different approaches for this idea have been proposed [AACdlO$^+$13a, AACdlO$^+$13b, BMKL12]. These approaches feature a central controller for a set of coordinated BSs, which introduces similar requirements in terms of data rate and latency to the backhaul network as CoMP.

One example for an SDN controller architecture for wireless coordination is the CROWD Controller Architecture (CCA), which I describe in detail in Section 8.2.

2.4 Application Layer

On the application layer, I focus on HTTP Live Streaming, which I introduce in Section 2.4.1. I give an overview of existing approaches for data rate prediction in Section 2.4.2.

2.4.1 HTTP Live Streaming

HTTP Live Streaming (HLS) is an HTTP-based multimedia streaming protocol that has been initially developed by Apple Inc. [App] and has been published as an IETF draft [PMA13]. It is, among others, available in the stock media players of Android [Goo] and iOS [App] operating systems and is available as an open-source implementation in the VLC player [Vid].

To stream a video using HLS, the video has to be encoded for it. This encoding is a CPU-intensive, one-time task. The video input is cut into independently playable segments with equal playback durations. Uniform Resource Locators (URLs) of these segments are then added to a playlist. An example of a typical HLS playlist is shown in Figure 2.5.

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-TARGETDURATION:10
#EXTINF:10,
http://hostname/high/001.ts
#EXTINF:10,
http://hostname/high/002.ts
#EXTINF:10,
http://hostname/high/003.ts
#EXTINF:10,
http://hostname/high/004.ts
#EXTINF:10,
http://hostname/high/005.ts
#EXTINF:10,
http://hostname/high/006.ts
#EXTINF:10,
http://hostname/high/007.ts
#EXTINF:10,
http://hostname/high/008.ts
#EXT-X-ENDLIST
```

Figure 2.5: Single variant HLS example with high-quality segments, each 10 seconds long

```
#EXTM3U
#EXT-X-STREAM-INF:BANDWIDTH=1000000
http://hostname/low/hls.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=1500000
http://hostname/med/hls.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=3000000
http://hostname/high/hls.m3u8
```

Figure 2.6: Multi-variant master playlist with three variants

HLS can provide multiple variants of a video. Each variant can be encoded using a different codec, bit-rate, or resolution. HLS players can switch between different variants for each segment because all segments have equal length and are independently playable. A separate playlist is created for each variant, in addition to a master playlist with links to all variant playlists. An example of a master playlist with three variants is shown in Figure 2.6. The master playlist contains parameters for each variant to enable HLS players to select the most appropriate one. For example, the required download data rate as `BANDWIDTH` in bit/s.

Playlists and segments are placed on a normal HTTP server. An HLS player only needs the URL to the HLS master playlist. From there, all variants and their segments can be accessed.

The standards for HLS do not specify how the download of segments should be organized, thus there is no standard download scheduling algorithm for HLS. I analyzed the source code of the HLS component of the VLC media player [Vid] resulting in three findings:

1. VLC has a fixed buffer it always tries to fill with video segments.
2. The quality selection is based on a measurement of the current connection speed.
3. The size of the buffer is measured in number of segments and not in storage space.

I use these findings to model HLS download scheduling in this thesis in Chapters 4 to 7.

2.4.2 Data Rate Prediction

Prediction of network traffic patterns in wired networks have been studied and appropriate algorithms exist [SL02]. However, for mobile access networks the physical layer exhibits a much larger variance as many factors influence the SINR that directly correlates with the achievable data rate. In spite of this, it has been shown that achievable rates can be predicted for different time scales and with different contextual information taken into account:

It can be observed that in many cases user movement follows recurring regular patterns (e.g., the way to work, public transportation networks), which leads

to the idea that such regular patterns are reflected in the achievable data rate which can then also be predicted. Song et al. [SQBB10] analyzed the entropy of over 50.000 user mobility patterns and concluded that human movement behavior shows highly regular patterns, which leads to a correct prediction of such patterns of up to 93 % of the investigated cases.

Yao et al. [YKH08] analyzed the feasibility of predicting available data rates based on information gathered during repeated trips in a range of eight months. The results are based on the analysis of the entropy as a measure for the uncertainty and are thus independent from a particular prediction technique. They show that there is no significant correlation between signals at different points of time during a single trip. But when the measurements of past trips are taken into account, accuracy of the data rate prediction increases significantly. Using the UE location as a context information improves accuracy even further.

The results from these studies strongly support the feasibility of data rate prediction in mobile access networks. They also indicate that the feasibility and accuracy of data rate prediction does not depend on a particular prediction technique. Thus I do not focus on any particular prediction technique in this thesis and introduce a generic model in Section 5.2.

3

State of the Art & Related Work on Anticipatory Download Scheduling

My approach for *anticipatory download scheduling* originates from the context of the "Smarter Phones and Networks" (SPAN) project, a cooperation with Alcatel Lucent Bell Labs Stuttgart, which I introduce in Section 3.1. After that I give an overview of related work in Section 3.2.

3.1 Smarter Phones and Networks

The main idea behind the "Smarter Phones and Networks" (SPAN) project was to develop a new technology called "Context-Aware Video Streaming", based on Alcatel Lucent Bell Labs' platform "Context-Aware Radio Access" (CARA). This technology makes use of location-based information, including road and network coverage maps and radio performance data, to enable algorithms that predict upcoming network performance. Based on this prediction, a video player application can download more video content to the user's device in areas with good wireless coverage before the user enters an area with poor wireless coverage. This results in a dramatic reduction in video playback interruptions and the quality of experience is greatly improved for the user. In the context of this project, I worked together with Frederic Beister and Stefan Valentin, whose work in the context of the project I introduce in this section.

Stefan Valentin focused on aspects related to wireless access. Together with H. Abou-Zeid and H.S. Hassanein [AzHV14] he presented the "Predictive Green Streaming" (PGS) optimization framework for using predicted wireless data rates to reduce the required wireless transmission time for mobile video streaming. The goal was to reduce the overall energy consumption of the mobile access network. In a simulation together with N. Barman [BV14], he investigated the prediction of future wireless data rates based on radio coverage maps in conjunction with street

maps. The results show that for a user moving between two street intersections the future data rates until the arrival at the next intersection can be predicted using radio coverage maps. He also investigated a wireless resource allocation scheme [Val14] based on prediction of future channel states to minimize required wireless resources for smooth video streaming. Finally, together with W. Bao [BV15], he developed an approach for anticipatory buffering of HTTP Live Streaming (HLS) segments based on observation of the current channel state and prediction of future channel states. The approach is based on Markov Decision Processes (MDPs) obtained from real traces of vehicular users.

Frederic Beister focused on the prediction of user behavior. He analyzed traces of video downloads from a large European mobile network operator to deduce patterns in the behavior of users and HLS video players with respect to mobile video streaming [BK14, Bei14]. Based on this analysis, he created Hidden Markov Models (HMMs) to predict the inter-download times of HLS video segments. Together with him, I also investigated the design of power consumption models for network equipment [BDAK14, DBK+12].

3.2 Related Work

Incorporating channel prediction into video streaming in mobile networks has been investigated in general [LdV13, KPS+06] and specifically for public transport scenarios [RVGH13, REV+12, YKH12, YKH11, YKH08]. None of these consider video quality selection *and* buffering, which results in a significantly less complex problem than my combined approach. Lu et al. [LdV13] propose an online algorithm for scheduling the transmissions of video segments and evaluate this approach using measured wireless capacity traces. Khan et al. [KPS+06] propose a cross-layer optimization strategy that jointly optimizes the application layer, data link layer, and physical layer to maximize user satisfaction with wireless video streaming. Riiser et al. [REV+12, RVGH13] analyze real-world measurement traces of the wireless data rate for users on popular public transport routes in Oslo, Norway, and propose a GPS-based bandwidth-lookup service in order to better predict near-future data rates and create a schedule for the video playout that takes future availability of data rates into account. Yao et al. [YKH12, YKH11, YKH08] investigate the predictability of future available data rates for commuting users and also propose an adaptive downloading scheme based on data rate maps.

Recent studies have also investigated the quality adaptation mechanism of HTTP video streaming [HJM13, LZG+14] in a *reactive* way, without the combination with data rate prediction. Huang et al. [HJM13] propose a buffering scheme based on the currently observed buffering behavior and not on the prediction of data rates. Li et al. [LZG+14] introduce a scheme for adapting the download behavior in HLS based on probing of the available data rate. Miller et al. [MQGW12, MATW15] investigated how the quality adaptation mechanism of HTTP video streaming can be adapted by measurements of the current TCP throughput. They use a simple moving average over the past and TCP throughput

as a short-term predictor for the future available throughput.

Approaches and results presented by Radhakrishnan et al. [RTN12, RN12] investigate the performance of Scalable Video Coding (SVC) [SMW07] in Long Term Evolution (LTE) networks. SVC is a video encoding technique which encodes a video as multiple subset video bitstreams. Such a subset video bitstream can be derived by dropping packets from a larger video to reduce the bandwidth required for the subset video bitstream. In SVC the quality adaptation mechanism [SMW07, OAJ14] is more complex than the simple selection of a quality from an existing set of video streams as in HLS.

Anticipatory approaches based on existing data rate traces have also been presented [HZM14, BHK13] that focus solely on the quality selection and do not incorporate the full download scheduling with the option to pre-buffer a variable number of segments. Hao et al. [HZM14] present a video streaming system that uses the GPS sensor in mobile phones to lookup future available data rates in a coverage map. Bokani et al. [BHK13] propose a data rate prediction scheme for wireless video streaming based on MDPs.

The measurements by Riiser et al. [RBV+12] and Müller et al. [MLT12] illustrate the bad performance of unmodified HLS in mobile networks, but do not include any prediction mechanism or cross-layer approach. Finally, a survey by Seufert et al. [SES+14] gives an overview of various approaches and improvements for HTTP video streaming, none of which incorporates any data rate prediction mechanism. In the following chapters I focus on filling this gap.

Finally, Hoßfeld et al. have investigated which factors impact the Quality of Experience (QoE) of users watching HTTP video streams. They conclude that 1) playback interruptions have the biggest negative impact on the QoE [HSH+11] and 2) switching between qualities of the video during playback has no negative effect as long as the video quality is high on average [HSSZ14, HSS+15]. They also argue that both metrics, playback interruptions and video quality, should be analyzed separately because a combined QoE metric is difficult to define. I use these findings to define the objectives for anticipatory download scheduling in the following chapters.

4

Anticipatory Download Scheduling with Perfect Prediction

In this chapter, I introduce the approach for anticipatory download scheduling based on HTTP Live Streaming (HLS). I assume that there is no limit or error on the prediction of future available data rates: precise predicted data rates for an unlimited future are always available to a scheduler. While this assumption is not necessarily realistic, it allows to create an optimal solution to establish a baseline for comparison with other solutions.

Based on the findings by Hoßfeld et al. [HSH+11, HSSZ14, HSS+15] described in Section 3.2 the objectives for anticipatory download scheduling are the following, sorted by their importance:

1. Minimize playback interruptions and avoid them completely if possible
2. Maximize video quality on average
3. Minimize buffering to avoid waste of wireless resources

After a problem description in Section 4.1, I formulate an optimal solution in Section 4.2 and introduce a heuristic algorithm in Section 4.3. For a comparison with the state of the art behavior of HLS players, I introduce greedy algorithms without prediction in Section 4.4. In Section 4.5, I show evaluation results for both solutions and compare them to the greedy algorithms.

4.1 Problem Description

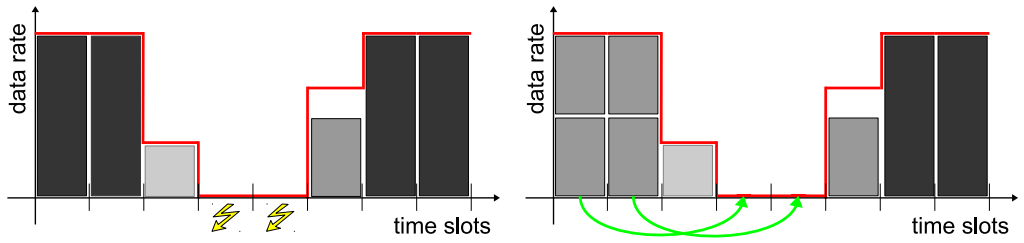
In Section 2.4.1, I have introduced the basic mechanisms of HLS, including segmentation of a video stream and the ability to encode a video in multiple video qualities that can be selected during the playback of the video. The segmentation implies that for uninterrupted playback, segment $i+1$ has to be downloaded entirely before segment i has been played to its end in the player application. HLS allows to *buffer* segments in the player application, i.e., download them before they are immediately needed for playback, to avoid playback interruptions. For the video quality, the video player application selects the quality level, for example, based on its measured downlink data rate.

By combining segment buffering and quality selection with the prediction of future available data rates the following scheduling decisions for downloading video segments have to be taken:

1. Downloading and buffering more segments in advance if a short period of bad channel quality is predicted for the near future.
2. Switching to a lower quality to download and buffering more segments in advance if a longer period of bad channel quality is predicted for the near future.

The advantage of such an anticipatory download schedule is shown in Figure 4.1. Darker shades for video segments indicate higher video quality and thus larger segment size. Since the duration of a segment is fixed, larger segments need a higher data rate to download. As a simplification I assume that a segment has to be downloaded completely within a single time slot. When downloading segments greedily without any prediction of future available data rates, as shown in Figure 4.1a, lack of available data rate in time slices 4 and 5 leads to a playback interruption because no segments are buffered in advance and the whole available data rate in time slices 1 to 3 is used to download segments with the maximum possible quality. In contrast to that, when the lack of available data rate in time

slices 4 and 5 is predicted and taken into account for the selection of segments to download, as depicted in Figure 4.1b, the playback is not interrupted because enough segments at a lower quality level are buffered in time slices 1 and 2.



(a) No download scheduling (b) Anticipatory download scheduling

Figure 4.1: Scheduling Example

With the model in this chapter, an anticipatory download schedule is created independently for each user, as the available data rates are given from the radio resource scheduler which takes care of the resource sharing between the users. Strictly speaking the optimization problem in the next section should thus be implemented without the consideration of multiple users. But the extension of anticipatory download scheduling for energy efficiency in Chapter 6 requires the consideration of multiple users. To keep both models compatible I already include the consideration of multiple users into the model in this chapter.

4.2 Optimal Solution

The Mixed Integer Quadratically Constrained Program (MIQCP) described in this section precisely specifies the previously illustrated scheduling problem and is amenable to an automatic solution. Before describing the actual MIQCP I describe the model assumptions.

4.2.1 Model Assumptions

I use a discrete time model, which means that time is represented as a discrete sequence of time slots t_i with equal lengths. For simplification, I assume that the length of every time slot t_i is equal to the HLS video segment length. Also, an HLS video segment is either downloaded completely within a single time slot or considered as not downloaded at all, i.e., HLS video segments cannot be downloaded across multiple time slots. The video download starts in the first time slot and ends in the last time slot. Thus, for the video to be played back without any interruptions every HLS video segment s_i has to be downloaded in time slot t_i at the latest. This means that segment s_i does not already have to be downloaded in time slot t_{i-1} which is a simplification to make the optimization problem more straightforward. Otherwise there would be no playback in time slot t_0 and no download in time slot $t_{|T|}$. If a segment s_i is not downloaded in or before time

slot t_i and the playback is interrupted, I quantify the *lateness* of segment s_i as the difference between time slot t_i and the time slot t_j when the segment was actually downloaded.

Downloads in each time slot are constrained by the available data rate of each user. Furthermore, each segment has a certain video quality level which determines the required data rate to download the segment with the selected quality level.

4.2.2 Mixed Integer Quadratically Constrained Program

Input parameters to the MIQCP are listed in Table 4.1 and decision variables in Table 4.2. Although I assume that the number of time slots T and the number of segments S are equal, the model is also capable of modeling different numbers of time slots and segments. The capacity per user is calculated as the capacity after a radio resource scheduler divides the available wireless resources and thus each $C_{u,t}$ is independent. The video qualities are defined in required capacity, thus $C_{u,t}$ and Q_i have the same unit.

Table 4.1: MIQCP input parameters

T	set of time slices, e.g., $\{0, 1, 2, 3\}$
S	set of segments to transfer, e.g., $\{0, 1, 2, 3\}$
U	set of users
$C_{u,t}$	capacity (data rate $\cdot$ segment length) for user u in time slot t
Q	set of video qualities e.g., $\{10, 20, 50\}$

Table 4.2: MIQCP variables

$d_{s,t,u} \in \{0, 1\}$	deliver segment s at time t to user u
$e_{s,u,q} \in \{0, 1\}$	deliver segment s for user u at quality q
$f_{s,u} \in \mathbb{N}$	quality for segment s for user u
$g_{s,t,u} \in \mathbb{N}$	quality for segment s for user u at time t
$l_{s,u} \in \mathbb{N}$	lateness per segment s for user u
$m_{s,u} \in \mathbb{N}$	aggregated lateness per segment s for user u after playback
$b_{t,u} \in \mathbb{N}$	number of buffered segments at time t for user u

First, it has to be ensured that each segment is downloaded exactly once (Eq. 4.1) and that exactly one quality level is selected for each segment (Eq. 4.2).

Of course multiple segments can be downloaded in one time slot.

$$\sum_{t \in T} d_{s,t,u} = 1, \forall s \in S, u \in U \tag{4.1}$$

$$\sum_{q \in Q} e_{s,u,q} = 1, \forall s \in S, u \in U \tag{4.2}$$

Then the selected quality for each segment (Eq. 4.3) and value for the decision variable that indicates when a segment with a selected quality level is actually downloaded (Eqs. 4.4 and 4.5) have to be calculated.

$$f_{s,u} = \sum_{q \in Q} e_{s,u,q} \cdot q, \forall s \in S, u \in U \tag{4.3}$$

$$g_{s,t,u} \geq f_{s,u} \cdot d_{s,t,u}, \forall s \in S, t \in T, u \in U \tag{4.4}$$

$$g_{s,t,u} \leq f_{s,u} \cdot d_{s,t,u}, \forall s \in S, t \in T, u \in U \tag{4.5}$$

Now the segment download has to be constrained by the per-user data rate (Eq. 4.6).

$$\sum_{s \in S} g_{s,t,u} \leq C_{u,t}, \forall t \in T, u \in U \tag{4.6}$$

The constraints specified so far only require each segment to be download eventually but do not consider playback interruptions caused by delayed segment downloads.

The quadratic nature of the problem, in contrast to being entirely linear, stems from the combination of quality selection and scheduling. It requires for each video segment s to multiply the integer decision variable for the selected quality $f_{s,u}$ with a binary decision variable $d_{s,t,u}$ to determine the scheduled time slice. This decision turns the problem from a linear one to a quadratically constrained problem. The problem retains its positive semidefinite nature and is thus still efficiently solvable by an optimizer.

To formulate the objective function additional constraints to handle playback interruptions and buffering of segments are required.

The lateness for each segment (Eq. 4.7) and the aggregated lateness after playback (Eq. 4.8) can now be calculated.

$$l_{s,u} = \sum_{t \in [s+1, \max(T))} d_{s,t,u} \cdot t, \forall s \in S, u \in U \tag{4.7}$$

$$m_{s,u} = \sum_{x \in [0, s+1)} l_{x,u}, \forall s \in S, u \in U \tag{4.8}$$

The number of buffered segments for each user u in time slot t also have to be taken into account. They are calculated by summing up the number of downloaded segments until t and subtracting t because up to time slot t, t segments have to

be played out (Eq. 4.9).

$$b_{t,u} = \left(\sum_{\substack{\forall x \in [0,t) \\ s \in S}} d_{s,x,u} \right) - t, \ \forall t \in T, u \in U \tag{4.9}$$

4.2.3 Objective

To formulate the objective function I define three weight factors: W_l for the lateness of video segments, W_q for the selected quality level of the video segments and W_b for the number of buffered segments. I formulate the objective function as

$$\begin{aligned} \text{minimize: } W_l \ \cdot & \sum_{s \in S, u \in U} l_{s,u} \\ - W_q \cdot & \sum_{s \in S, u \in U} f_{s,u} \\ + W_b \cdot & \sum_{t \in T, u \in U} b_{t,u} \end{aligned} \tag{4.10}$$

This formulation allows both to trade-off the metrics lateness, video quality, and buffer usage as well as to define a lexicographical ordering of these metrics, which I will do for my evaluation. As an alternative to this objective function it is also possible to set fixed limits to one or two of the metrics and to maximize or minimize the remaining ones, deriving a corresponding Pareto front.

With respect to the Quality of Experience (QoE) for the users and based on the findings by Hoßfeld et al. [HSH+11, HSSZ14, HSS+15] described in Section 3.2, I use a lexicographical order of 1) minimizing lateness, 2) maximizing video quality, and 3) minimizing buffer usage. The precise weights depend on the number of users and segments, but the following inequality has to hold: $W_l >> W_q >> W_b$. I also use this lexicographical order for designing heuristic algorithms for which I use the optimization problem as a baseline for evaluation.

4.2.4 Complexity

The problem underlying the MIQCP is a combination of two knapsack problem variants [KPP04]:

The decision in which time slice to download each segment corresponds to the *multiple knapsacks* problem: Each time slice is one of the multiple knapsacks and the reward for putting a segment into a knapsack corresponds to whether the segment is downloaded too early, on time or too late. If this concept is transformed to a suitable reward function, a Polynomial-Time Approximation Scheme (PTAS) exists to approximately solve this problem in polynomial time [CK00].

The decision which quality to use for each segment corresponds to the *multiple choice knapsack* problem, where the choice of items is limited to a certain sets

of items. This corresponds to the different quality levels in my combined problem. The multiple choice knapsack problem can also be solved approximatively in polynomial time with time $O(n \cdot \log(n) + \frac{m \cdot n}{\epsilon})$ and space $O(n + \frac{m^2}{\epsilon})$ where n is the number of items and m the number of sets to choose from [Law79].

I cannot simply divide the scheduling problem into completely separate instances of both knapsack problems in a way to apply both approximations separately. Instead, both PTAS approaches would have to be combined into a new approximation to solve our scheduling problem. Since both PTAS approaches alone are already very complex and this is only the baseline case with perfect prediction, I developed a heuristic algorithm instead of a fixed approximation algorithm.

4.3 Fill Algorithm

Consistent with the optimization problem, the FILL algorithm is an offline scheduler. Data rates for all time slots are known in advance and the result of the algorithm is a complete schedule for all users over all time slots. The assumptions are the same as for the optimization problem.

The FILL algorithm takes available data rate for each user in each time slot as its input parameter and iterates over all time slots to fill the buffer with video segments independently for all users. Algorithm 4.1 shows this basic structure of the algorithm.

Algorithm 4.1 FILLSCHEDULER(U, T, Q)

```
1: // users U, times T, qualities Q
2: for all u ∈ U do // schedule all users
3:   C ← PREDICTUSERRATES(u) // from channel prediction
4:   s ← 0 // initialize counter for scheduled segments
5:   for all t ∈ [0..|T|] do // schedule all time slots/segments
6:     s ← s + SCHEDULESEGMENT(u, t, s, Q, C)
7:   end for
8: end for
```

The function PREDICTEUSERRATES(u) returns the predicted data rates for a user for all time slots based on the underlying radio resource scheduler.

The control flow of SCHEDULESEGMENT (Algorithm 4.2) is illustrated in Figure 4.2. For each time slot there are two different operations, depending on the available data rate in that time slot: If there is enough data rate to download a new segment in the currently examined time slot (Algorithm 4.2, lines 3 and 4), the FILL algorithm schedules this video segment at maximum possible quality. This behavior ensures a minimum number of segments in the buffer as long as there is no need for buffering more segments for future time slots with insufficient data rate. If, during the iteration, the predicted data rate in some time slot t does not suffice to download a new video segment (Algorithm 4.2, lines 6 to 22) even at the lowest video quality level, the FILL algorithm has to change the schedule for one

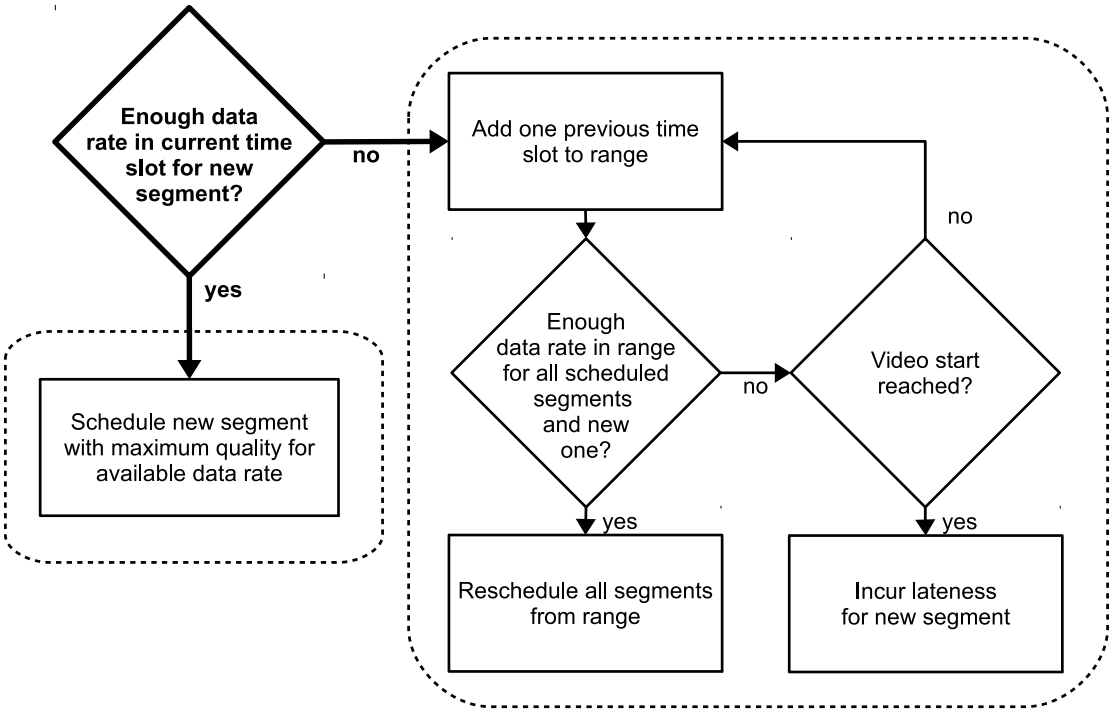


Figure 4.2: Flowchart for Fill Scheduler, specifically Algorithm 4.2

or more *previous* time slots to download and buffer a video segment before time slot t with insufficient data rate.

This part of the algorithm, outlined in Algorithm 4.2, requires the following helper functions:

- getBestQuality(Q, c)
  Returns the best downloadable quality (out of Q) for a segment with predicted available data rate c, or FALSE if the data rate is insufficient even for the lowest quality

- getBestQualityRange(Q, n, c)
  Returns the best possible quality (out of Q) at which n segments can be downloaded with predicted available data rate c, or FALSE if there is not enough data rate to download all n segments even in the lowest quality

- getSegmentsForQuality(q, c)
  Returns the number of downloadable segments with quality q and available data rate c

- schedule(u, s, t, q)
  Schedule the download of segment s for user u at time t with quality q

From time slot t where downloading of a full segment was not possible, the algorithm goes back time slot by time slot. In the revisited time slots, it downgrades the video quality of the segments, freeing up capacity to enable downloading of the segment that has to be played out in time slot t. It can change the scheduled

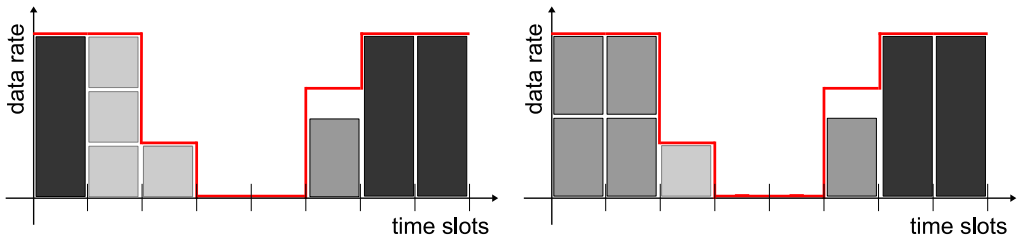
Algorithm 4.2 scheduleSegment(u, t, s, Q, C)

```
1: q ← getBestQuality(Q, C[t])
2: if q ≠ false then // enough capacity in current time slot for new segment?
3:   schedule(u, s, t, q) // schedule new segment with maximum quality for available
     data rate
4:   return 1
5: else // even lowest quality not feasible in time slot t
6:   for all g ∈ [t..0] do
7:     // enough capacity in range [g..t] for all scheduled segments and new one?
8:     if getBestQualityRange(Q, t − g + 1, Σ_{i=g}^{t} C[i]) ≠ false then // going back to
       g provides enough data rate
9:       q ← getBestQualityRange(Q, t − g + 1, C[g...t])
10:      p ← 0
11:      for all r ∈ [g..t] do // reschedule all segments from range
12:        n ← getSegmentsForQuality(q, C[r])
13:        for all v ∈ [(g + p)..(g + p + n)] do
14:          schedule(u, v, r, q)
15:        end for
16:        p ← p + n
17:      end for
18:      return 1
19:    end if
20:  end for // video start reached
21:  return 0 // incur lateness for new segment
22: end if
```

download times of earlier segments in order to fit more segments into time slots. Once a range of time slots is found where all segments including the one to be played out in time slot t fit in (at reduced quality), computation of the schedule up to time slot t is complete. This schedule serves as the basis to plan downloading the segment for time slot $t + 1$ in the next iteration.

For example, consider the situation in Figure 4.3a. The rectangles show the video segments with different qualities, indicated by their shading, and the solid line above the rectangles indicates the available data rate. There is not enough data rate in the fourth and fifth time slots to download a video segment, but in the second time slot there is enough data rate to download three segments. So the algorithm goes back to the third time slot and determines that only going back to the third time slot is not sufficient. Then it also move back to the second time slot and resolves the lack of data rate in the fourth and fifth time slots so the video can be played back uninterruptedly.

The downside of the Fill algorithm is the fact that the reduction of playback interruptions results in a significant reduction of the video quality level. By comparing the example schedule from the Fill algorithm in Figure 4.3a with the schedule generated from the optimization problem in Figure 4.3b this behavior becomes obvious: the Fill algorithm goes back to the second time slot, and there



(a) Example for FILL Algorithm (b) Example for MIQCP Schedule

Figure 4.3: Coordinated Multipoint Transmission and Reception

it can resolve the lack of data rate in the fourth and fifth time slots by downloading the segments in the lowest video quality, whereas going back to the first time slot and downloading the segments with a medium video quality level provides a better average video quality level. In this toy example, one could argue to allow the FILL algorithm to go back a number of additional time slots to fix that issue. But in a real scenario there is no way to reasonably limit such a number of additional time slots to consider, and it would essentially turn the algorithm into an exhaustive search algorithm.

4.4 Greedy Algorithms

I this section, I introduce two greedy scheduling algorithms, that illustrate the behavior of standard HLS player applications as a reference of the evaluation. Since there is no specified default behavior for HLS player applications, I introduce two different greedy behaviors, each with its specific mode of operation: 1) keeping a full buffer *or* 2) maintaining a high video quality.

Both greedy scheduling algorithms take the available data rate for each user in each time slot and a maximum buffer size as their input. Based on their respective mode of operation, they iterate over all time slots and decide which segments to download to fill the buffer with video segments. In each time slot, they consider the currently available data rate, the current number of buffered segments, and the quality levels of the segments not yet scheduled.

These algorithms cannot change the maximum buffer size and both fill the buffer according to their respective mode of operation. This can result in unnecessary buffering if enough data rate is available to play the video without buffering, or unwanted playback interruptions if the chosen buffer size is not big enough to continue playback in phases of insufficient data rate.

4.4.1 BufferFirst Algorithm

The mode of operation of the BUFFERFIRST algorithm is to fill the entire buffer with video segments. If the buffer is not completely full in a time slot, the algorithm initially schedules the download of the maximum possible number of

segments at the lowest quality supported by the currently available data rate and free buffer space. If there is also enough data rate available to download segments in higher quality levels, it afterwards increases the quality for the scheduled segments to create a final schedule for each time slot. Thus, the algorithm never decides to download fewer segments to increase the quality.

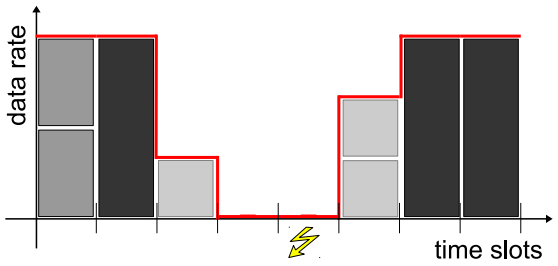


Figure 4.4: Example for BufferFirst Algorithm

Figure 4.4 shows an example for the BufferFirst algorithm with a maximum buffer size of *two* segments. The red line indicates the available data rate in each time slot. The buffer is filled with segments of medium quality in the first time slot. In the second and third time slots, one segment is downloaded to fill the buffer again. With this schedule the video playback will not be interrupted in the fourth time slot because there is still one segment in the buffer. But the video playback is interrupted in the fifth time slot.

4.4.2 QualityFirst Algorithm

The mode of operation of the QualityFirst algorithm is to download segments with the highest quality possible. If the buffer is not completely full in a time slot, the algorithm schedules the download of new segments at the maximum possible quality depending on the currently available data rate. If there is still free buffer space and data rate it continues to schedule downloading further segments.

As a consequence, this algorithm favors downloading segments at higher video quality levels instead of buffering segments.

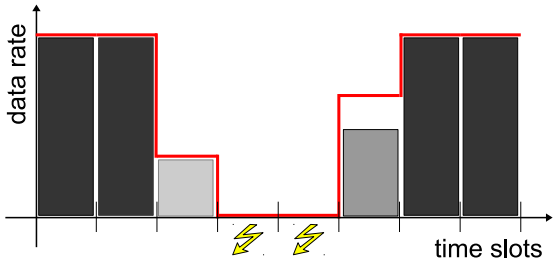


Figure 4.5: Example for QualityFirst Algorithm

Figure 4.5 shows an example for the QualityFirst algorithm with a maximum buffer size of *two* segments. As the algorithm favors to download segments with high quality levels before filling the buffer, there are no segments in the buffer to avoid a playback interruption in the fourth and fifth time slot.

4.5 Evaluation

With the MIQCP, the Fill algorithm and the two greedy algorithms, I can both compare the optimal solution to the state of the art, represented by the greedy algorithms, and compare the performance of the heuristic Fill algorithm to the optimal solution.

I first introduce the simulation scenario in Section 4.5.1 and present the evaluation results in Section 4.5.2.

4.5.1 Scenario

The basic structure for the evaluation scenario is a line of Base Stations (BSs) with fixed, equal distance. The users are moving along the BSs through the scenario from the first BS to the last BS as illustrated in Figure 4.6.

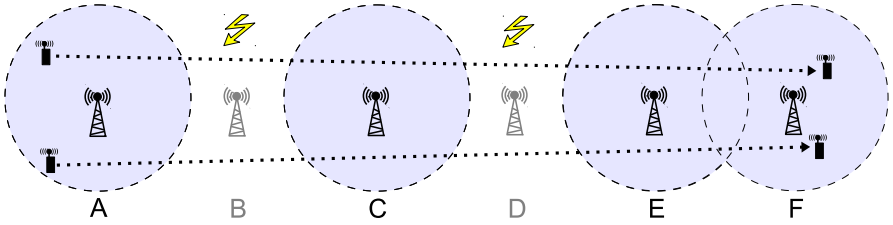


Figure 4.6: Evaluation scenario

To decrease the available data rate and to create the need for buffering, I remove BSs from the scenario, as illustrated by BSs B and D. The more BSs are removed, the more gaps without any available data rate occur and the more segments have to be buffered to avoid playback interruptions. The users all move as a group from the first BS to the last BS (e.g., a public transport scenario).

The wireless links are modeled according to the 3GPP LTE model (Section 2.2.1, [3GP09]). The BSs are placed equidistantly with an inter-site distance of 1500 meters, which is slightly larger than a normal urban scenario in order to augment the effects resulting from removing cells. I consider four active users in the scenario, to maintain comparability between the simulation results and the testbed measurements in Section 7.3.

For the channel capacity I assume an asymptotically error-free communication channel, modeled by the Shannon equation with the following parameters: 10 MHz bandwidth, 46 dBm transmission power, isotropic antennas with 0 dB gain, -174 dBm/Hz noise power spectral density and -149 dBm/Hz average interference. The maximum data rate for a BS is limited to 30 Mbit/s to account

for the small number of users in the scenario. The allocation of data rates to the users in each time slot is determined by a wireless resource scheduler, which is, in this case, a simple proportional fair scheduler executed independently per time slot. For each time slot, a user is associated with that BS that provides the best Signal to Interference and Noise Ratio (SINR).

The maximum buffer size for the greedy scheduling algorithms is set to 3 segments, which corresponds to the default setting for VLC on Android.

The video quality levels and the corresponding required data rates are taken from a HLS test video I have generated from the clip "Tears of Steel"[1]. The resulting segment sizes for the three video quality levels are 1.413 MB (low), 2.951 MB (medium) and 3.613 MB (high). As the real file size of all segments varies slightly by a few hundred kilobytes due to the video encoding, we use the maximum size over all generated video segments in one video quality level as the parameter for the scheduling algorithms. The segment length is set to 10 seconds, corresponding to the recommended value in the HLS standard. The video has a total length of 44 segments.

The scenario consists of a 44 BSs. The number of removed BSs varies from 0 to 20, which means that in the worst case half of all BSs are removed. The removed BSs are selected uniformly, but the first and last 2 of the 44 BSs are never removed to avoid side effects. Removing more base stations yields infeasible scenarios for the MIQCP because some segments can never be downloaded.

The weights for the MIQCP objective function, as described in Section 4.2, are set to enforce the following lexicographical order: minimize lateness before maximizing quality and before minimizing buffering ($W_l = 440$, $W_q = 10$, $W_b = 1$).

4.5.2 Results

I evaluate three different metrics: the average downloaded video quality level in MB per segment, the lateness of segments in seconds averaged over all users as an indicator for playback interruptions and the average buffer level in segments. All plots are based on multiple simulation runs and show confidence intervals at 95% confidence level; small intervals might be covered by the plot markers.

The simulation results for the average video quality are shown in Figure 4.7. The dashed lines indicate the reference value of the high, medium and low video quality levels. MIQCP delivers the overall highest video quality level, which decreases only slightly once more than 10 BSs are removed from the scenario. This indicates that MIQCP can exploit the available data rate in order to deliver and buffer high quality segments whenever possible. The QualityFirst algorithm delivers the overall second highest video quality level, which is only slightly less than the one from the MIQCP. This corresponds to the expected behavior of the QualityFirst algorithm. The BufferFirst algorithm exhibits the opposite behavior and delivers the overall lowest video quality level, which also corresponds to the expected behavior. The Fill algorithm provides the same high video quality level as the MIQCP when only a small number of BSs is removed and enough

[1] `https://mango.blender.org/`

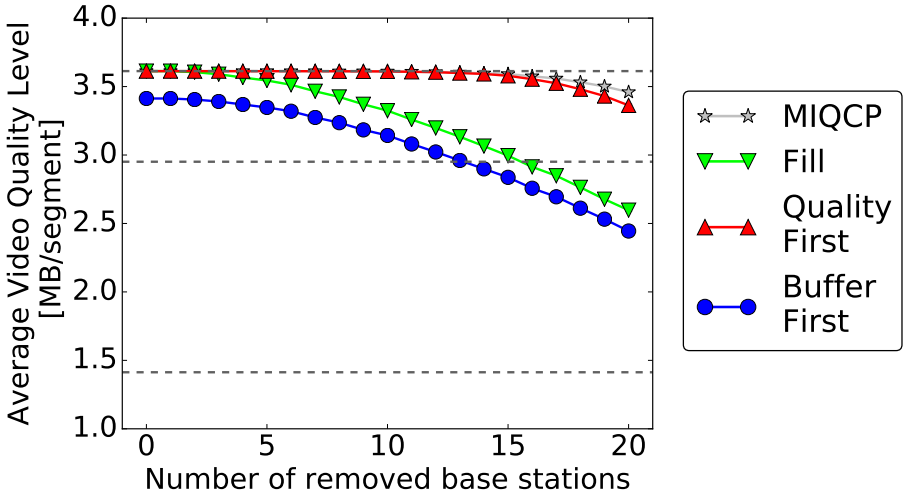


Figure 4.7: Simulation results: average quality

data rate is available. When more BSs are removed, the delivered video quality level from the Fill scheduler decreases but is still higher compared to the BufferFirst algorithm.

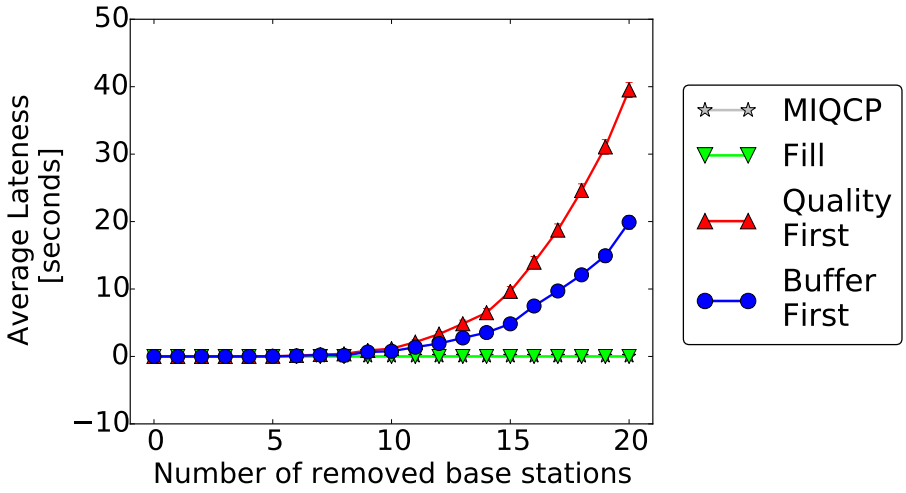


Figure 4.8: Simulation results: average lateness

Figure 4.8 shows the results for the average lateness over all users in the simulation. MIQCP and the Fill algorithm never download segments too late. For both the QualityFirst and BufferFirst algorithms lateness increases when more than 10 BSs are removed. Because of the mode of operation to download segments in higher quality levels instead of buffering more segments, the QualityFirst algorithm incurs the highest lateness.

The simulation results for the average buffer fill level are shown in Figure 4.9. Both greedy scheduling algorithms always try to fill their buffer up to the maxi-

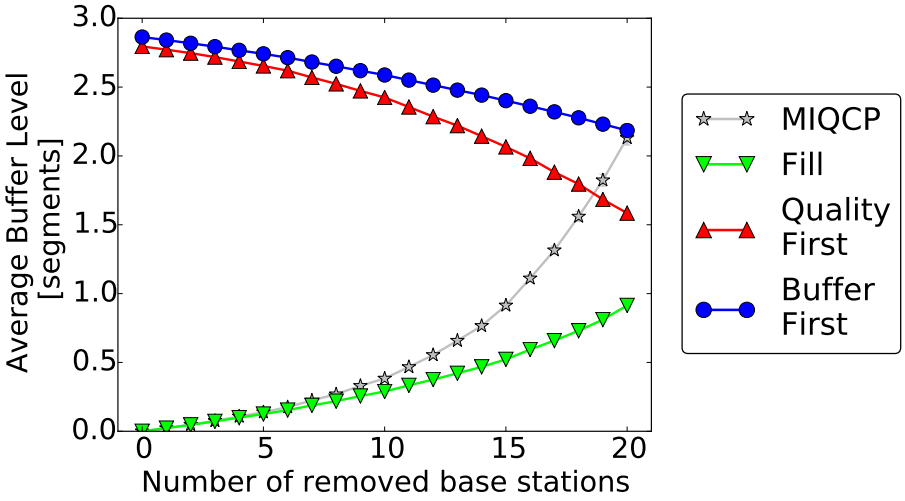


Figure 4.9: Simulation results: average buffer level

mum buffer level of 3 segments. Because the greedy scheduling algorithms have no mechanism to reduce buffer usage, buffer levels only decrease when there is not enough available data rate to fill the buffer entirely. This happens when more and more BSs are removed from the scenario. The MIQCP and the FILL scheduler are designed to minimize buffer usage where possible, thus both start off with little buffering and only increase the buffer level as more and more BSs are removed from the scenario. After removing more than 10 BSs from the scenario the MIQCP uses more buffer space than the FILL algorithm. This is caused by the preference of the MIQCP objective function to download segments with a higher video quality level rather than minimizing the buffer level. The FILL algorithm, on the other hand, will switch to lower video quality levels instead of buffering more segments.

4.5.3 Algorithm Running Times

Figure 4.10 shows the running times for the different scheduling algorithms on the simulation scenario. All schedulers are executed using a single Intel Xeon X5650 CPU at 2.67 Ghz.

Solving the MIQCP takes around 2.5 orders of magnitude longer than running the FILL algorithm. The running time of the FILL algorithm does not change significantly when more base stations are removed from the scenario. The running time of the greedy scheduling algorithms is less than the running time of the FILL algorithm because they are less complex than the FILL algorithm. Their running time slightly decreases as more base stations are removed from the scenario because with less available data rate, the greedy algorithms have to make fewer decisions on how to use the available data rate.

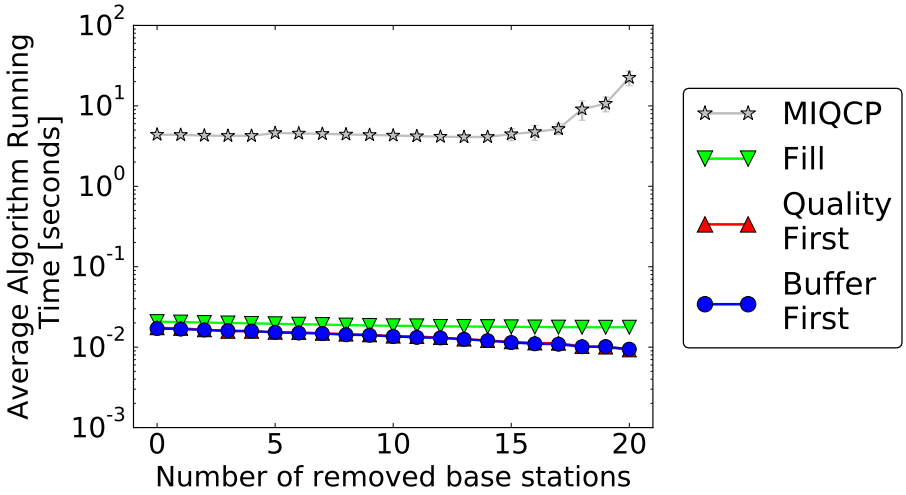


Figure 4.10: Algorithm running times

4.6 Summary

In this chapter I have described the anticipatory download scheduling problem with perfect prediction of available data rates. I have used this variant of the problem to establish a baseline case by formulating a Mixed Integer Quadratically Constrained Program (MIQCP) to solve the problem. I have also described how current state of the art HLS players schedule their downloads without prediction of future available data rates, to compare my solutions to a real-world case. Based on the behavior of the MIQCP I have implemented a heuristic algorithm to solve the problem.

The evaluation results clearly indicate that incorporating the prediction of future available data rates into the download scheduling essentially eliminates playback interruptions in the investigated scenario. With the solution from the MIQCP also the video quality is not decreased, whereas the heuristic algorithm reduces the video quality to avoid playback interruptions.

After establishing the baseline case in this chapter, I investigate anticipatory download scheduling with uncertain prediction of future available data rates in the next chapter.

5

Anticipatory Download Scheduling with Uncertain Prediction

In this chapter I describe how the approach for anticipatory download scheduling based on HTTP Live Streaming (HLS) can also be implemented with erroneous prediction of future available data rates. Here I assume that (a) data rate prediction is only available for a limited number of future time slots and (b) predicted data rates can be subject to errors. I call such erroneous predictions *uncertain*.

Together with Johannes Blobel, I have first created a generic predictor (Section 5.2) to pass information about the uncertainty of predictions to a scheduler. We then evaluated how the algorithms based on perfect prediction behave if used together with uncertain predictions (Section 5.3) and developed a new prediction-error-aware-scheduling algorithm (Section 5.4) that takes the uncertainty of predictions into account. We finally evaluated the new algorithm (Section 5.5) to show how anticipatory download scheduling is also feasible with uncertain predictions.

5.1 Problem Description

The basic underlying problem is the same as before in Chapter 4, but the scope of scheduling decisions is inherently different: When perfect prediction of future data rates is available, the problem to solve is to establish a download schedule for a whole scenario, i.e., the whole duration of a video playout. With uncertain prediction of future data rates, the problem to solve is to decide what do download in the current time slot and to take this decision in every time slot during the playout of a video because the prediction information is updated in every time slot. A scheduler also has to take the accuracy of predicted data rates into account.

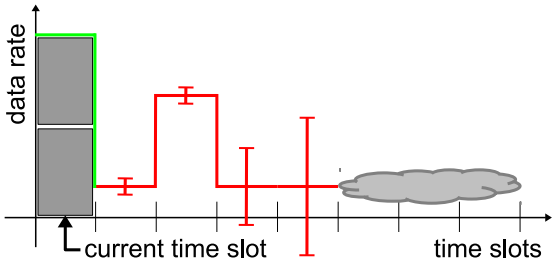


Figure 5.1: Example for scheduling with uncertain prediction

Figure 5.1 illustrates how a scheduler with uncertain prediction works. For the current time slot, the exact available data rate is known to the scheduler. For the next two time slots a predicted data rate with low probability of a prediction error (indicated by the error bars) is available. For the third and fourth future time slots, there are predictions of data rates but with a high probability of prediction error. For the following time slots, no predicted data rates are available to the scheduler. The scheduler now decides to trust the predictions for the next two future time slots, therefore it downloads and buffers a segment for the second time slot. Because the predictions for the fourth and fifth time slot are too uncertain, the scheduler does not trust them and does not schedule the download of additional segments yet.

5.2 Generic Predictor

In this section I present a generic error model for data rate prediction that can be used to implement a generic predictor to simulate the behavior of a real data rate predictor based on one of the introduced prediction techniques (see Section 2.4.2).

To evaluate the performance of anticipatory scheduling algorithms with uncertain predictions, it would of course be possible to just implement one of the prediction techniques introduced in Section 2.4.2. But the accuracy of these techniques highly depends on the amount and quality of used training data. Without sufficient training data the performance of an otherwise accurate algorithm can

significantly be degraded [GUR+13]. As it is known from machine learning techniques, the sets of training and validation data should be distinct from each other in order to receive valid results [Mit02]. In a simulation environment the training data would be generated from the same scenarios that the predictor is evaluated with.

To avoid this problem, we have implemented a generic predictor which does not predict future data rate based on any machine learning technique. Instead, predicted data rates are distributed to represent the predicted values from a real predictor. Thus, the uncertainties follow a similar distribution.

5.2.1 Stochastic Model of Prediction Errors

Bui et al. [BMW14, BW14] showed that data rate prediction errors can be modeled using Gaussian random walks. The main source of prediction errors of category 2 and 1 (see Section 2.4.2) are uncertainties in the system parameters, above all user distance to a Base Station (BS) resulting from the user's location. To obtain error sequences, they simulated a user's movement in a scenario in which BSs are randomly placed. For these scenarios they trained an Autoregressive Moving Average (ARMA) model to retrieve predicted values and the corresponding error sequences.

Then, they derived a model that describes data rate prediction errors as a normally distributed random variable with zero mean. The standard deviation of the error distribution σ was formulated as:

$$\sigma_k^2(s, T_s) = A(s, T_s)k + B(s, T_s) \tag{5.1}$$

where s is the user's speed, T_s the sampling period, k the prediction horizon, i.e., how far in the future the data rates are predicted. A and B are linear functions $A(s, T_s) = A_1 \cdot s \cdot T_s + A_2$ and $B(s, T_s) = B_1 \cdot s \cdot T_s + B_2$ whose coefficients have been fitted by minimizing the least square error between the model and the data. Results from Bui et al. [BMW14, BW14] show that distributions of the stochastic model and of the generated error sequences from the ARMA predictor are in fact identical. With the fitted coefficients the real errors could be matched closely.

5.2.2 Implementation

Based on the stochastic model, we have implemented a generic predictor to simulate the behavior of a real predictor by adding a Gaussian error to the actual data rate of a user. Because the user's speed and the sampling period are fixed throughout the simulations, the two functions A and B (Equation 5.1) can be replaced by constants: We replace A by a the error factor e to parameterize the accuracy of the model and replace B by 0 because we assume a perfect knowledge of the current available data rate (i.e., there is no intrinsic randomness in the prediction as assumed by Bui et al. [BW14]).

The error factor e represents the accuracy of the simulated prediction algorithm and the simulated training data. A small factor represent a good and accurate

prediction, a large factor a bad and uncertain prediction. The predictor's accuracy depends on the error factor e and the prediction horizon (i.e., how far into the future a prediction is made). According to this and Equation 5.1 from the stochastic model, the predicted data rate b of a user u at time slot t_i has a standard deviation of

$$\sigma_{u,t} = \sqrt{e \cdot (t_i - t_0))} \tag{5.2}$$

where t_0 is the time slot in which the predictor is queried.

The generic predictor can also be used to generate a prediction for the complete duration of the simulation. This ensures that the predicted values can also be given to the schedulers for perfect prediction that require knowledge of predicted data rates for the whole scenario.

5.3 Evaluation of Perfect Prediction Algorithms with Uncertain Predictions

In this section I present an evaluation of the FILL scheduler and the greedy schedulers (Chapter 4) with *uncertain predictions* using the generic predictor from the previous section.

5.3.1 Scenario

The evaluation scenario consists of a line of BSs similar to Section 4.5. In contrast to the previous evaluation in Section 4.5, here we use the GreenTouch radio model (Section 2.2.2). The scenario has 25 equidistantly placed BSs, 12 of which are removed uniformly at random to create gaps. The video duration is 30 minutes with a segment length of 10 seconds.

Since the schedulers for perfect prediction from Chapter 4 need predicted data rates for the whole scenario, data rates are calculated using the generic predictor from Section 5.2 which is queried every k-th time slot.

5.3.2 Results

Simulation results in Figure 5.2 show the influence of introducing an error factor e and the prediction horizon k on the existing schedulers. All plots are based on multiple simulation runs and show confidence intervals at 95% confidence level; small intervals might be covered by the plot markers.

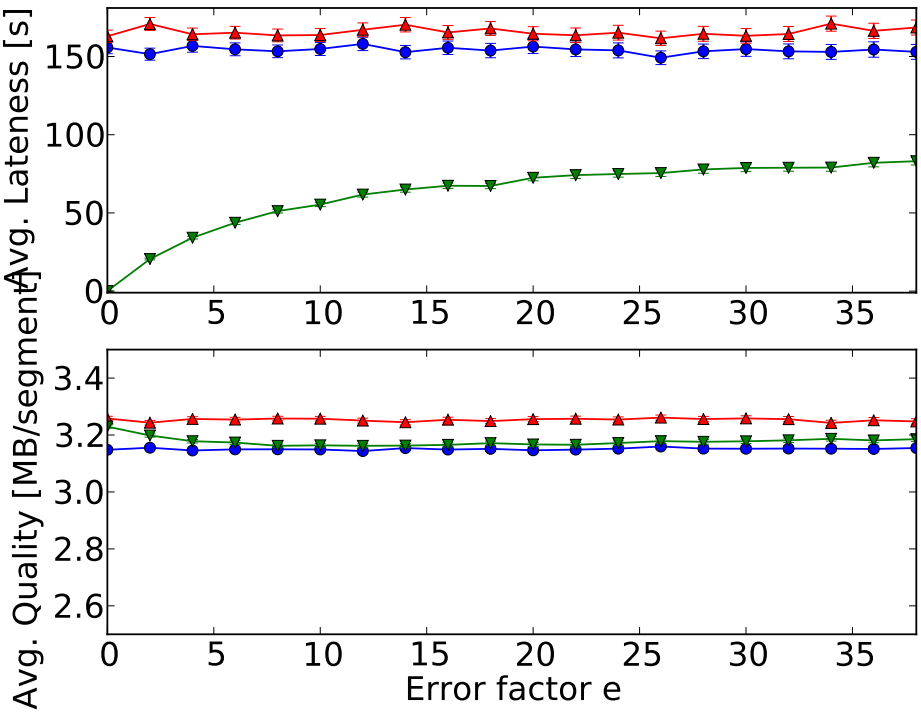
The greedy schedulers do not take anticipated data rates into account. Hence, the added error has no influence on the performance of these schedulers. Because 12 BSs are removed from the scenario, the greedy schedulers with a maximum buffer size of 3 segments incur a high lateness. Consistent with the previous results in Section 4.5, the QUALITYFIRST scheduler delivers a slightly better average video quality and slightly worse lateness.

As shown in Figure 5.2a, with a perfect prediction ($e = 0$), the FILL scheduler achieves much better lateness than the greedy schedulers, which is consistent with

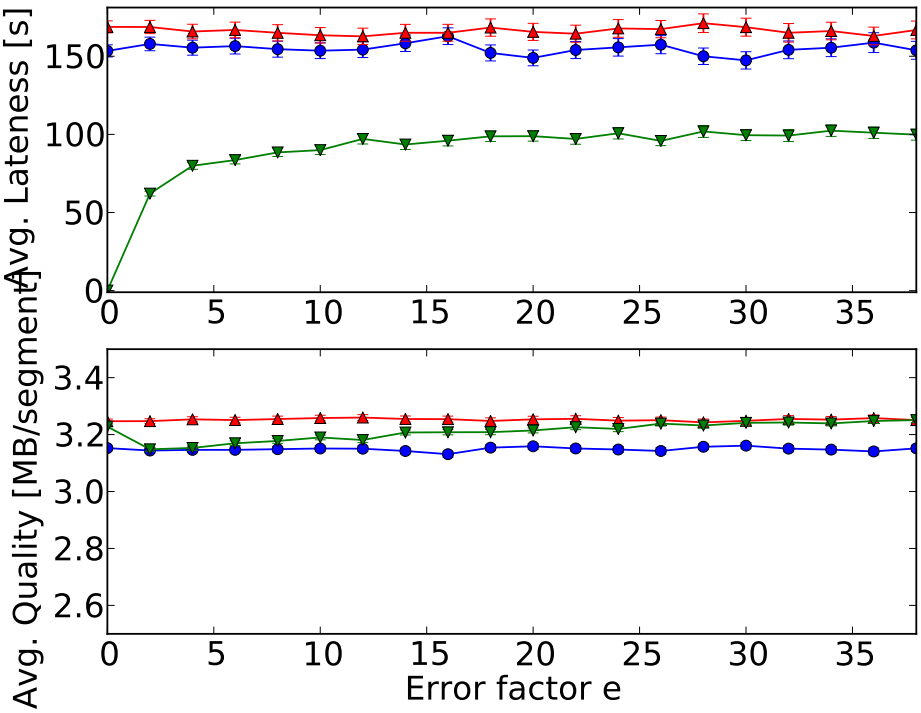
the previous results. But as the error factor is increased, the lateness also increases significantly.As before, average video quality is not affected by the prediction errors and consistent with the previous results from Section 4.5. This is due to the fact that the wrong decisions based on the prediction errors only delay the download of segments and do not change the quality selected by the scheduling algorithm.

The size of the prediction horizon k also influences the performance as shown in Figure 5.2b. The later the prediction horizon is (i.e., the less frequent the prediction is queried), the steeper is the increase of lateness for the FILL scheduler. This is due to the fact that with fewer queries of the predictor, the data rates that lie further in the future become more and more inaccurate.

The results clearly indicate that the existing FILL scheduler does not work with uncertain predictions. Thus we introduce a new prediction-error-aware scheduler in the next section.



(a) $k = 3$



(b) $k = 12$

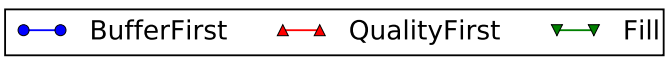


Figure 5.2: Evaluation of perfect prediction algorithms with uncertain predictions

5.4 Plan Algorithm

To make anticipatory download scheduling work in a scenario with uncertain predictions it is necessary to implement a scheduler that can handle prediction errors. The generic predictor that I introduced in Section 5.2 not only yields predicted data rate values but also the standard deviation of this value. This additional information about the accuracy of the prediction can be used by a scheduler to determine if it should consider or discard predicted values. The real future data rate cannot be determined with the knowledge of the standard deviation; it is only an indicator of the accuracy of the prediction. Without any additional information the scheduler would just have to guess the accuracy of the prediction and would have no substantial information to base its decisions on.

The design of this algorithm is based on the following considerations:

- The current data rate is more important than future values and should thus have a greater influence on the scheduling decisions than future predicted values.

- Predicted values with high standard deviation should have a smaller influence than predicted values with low standard deviation.

- When there is sufficient predicted data rate in the future, only the necessary segments should be downloaded and no additional segments should be buffered.

The scheduling algorithm is executed for every time slot to determine how many segments should be downloaded in which quality in this time slot. The algorithm is divided into four phases:

- *Phase 1: Dynamic Buffer*
  Determine the number of necessary segments

- *Phase 2: Summed Capacity*
  Calculate the weighted summed capacity (data rate over time)

- *Phase 3: Schedule Plan*
  Create a preliminary schedule plan for the next h time slots

- *Phase 4: Implementation of Plan*
  Schedule as much as possible from the preliminary plan in the current time slot

The basic structure of the algorithm is outlined as a flow chart in Figure 5.3 and the pseudo code is listed in Algorithm 5.1. Input variables to this algorithm are listed in Table 5.1.

The four phases of the algorithm are explained below in more detail.

Algorithm 5.1 PLAN$(t, k, s, c_c, C_{pred}, D_{pred}, Q)$

// Phase 1: Dynamic Buffer
$r \leftarrow 1$
$low \leftarrow$ **false**
for all $c_i \in C_{pred}, d_i \in D_{pred}$ **do**
  if $\max(c_i - 1.4 \cdot d_i, 0) < \min(Q)$ **then**
    $r \leftarrow r + 1$
    $low \leftarrow$ **true**
  end if
  if $\max(c_i - 1.4 \cdot d_i, 0) > \max(Q)$ **and** $low =$ **true then**
    $h \leftarrow x$
    break // End of low phase
  end if
end for
$n \leftarrow t + r - s$ // subtract already downloaded segments
// Phase 2: Summed Capacity
$C_s = c_c + \sum_{i=0}^{i \leq h} \max(c_i - 1.4 \cdot d_i, 0)$
with $c_i \in C_{pred}, d_i \in D_{pred}$
// Phase 3: Schedule PLAN
$P \leftarrow$ BESTQUALITIES(C_s, n, Q)
// Phase 4: Implementation of PLAN
while $\sum_{q_i \in P} q_i > c_c$ **do**
  REMOVESEGMENT(P)
end while
for $0 \leq i < |P|$ **do**
  SCHEDULE$(s + i, P_i)$
end for

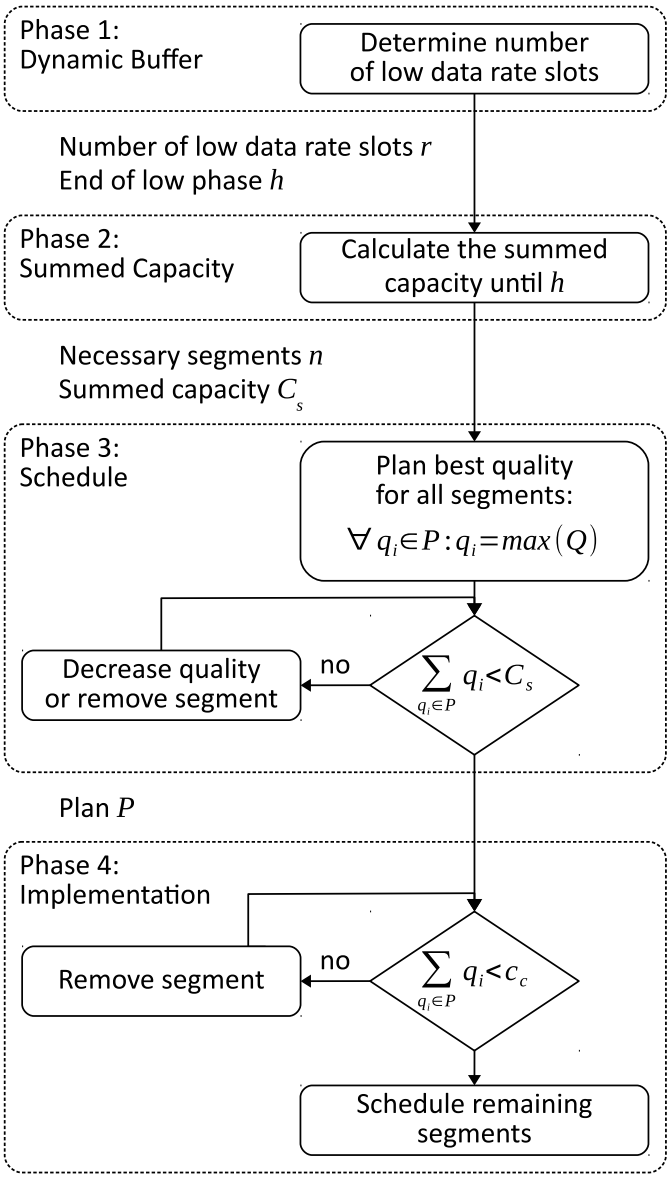


Figure 5.3: Flowchart of the PLAN algorithm

Variable	Description
t	Current time slot
k	Prediction horizon
s	Number of already loaded segments
c_c	Current capacity (data rate · time slot length)
C_{pred}	List of predicted future capacities (data rate · time slot length)
D_{pred}	List of corresponding standard deviations
Q	List of available video qualities (required capacity per segment)

Table 5.1: Input variables of PLAN, Algorithm 5.1

Phase 1: Dynamic Buffer The first step of the algorithm is to determine the number of segments n that should be downloaded in the current time slot to ensure an uninterrupted video playback. To do so it counts the number of future time slots r within the prediction horizon k, for which the predicted capacity c_i minus the standard deviation d_i is less than the lowest quality $\min(Q)$ (lines 1 to 12 in Algorithm 5.1). These time slots are called *low* time slots. Using $c_i - 1.4 \cdot d_i$ to estimate the real capacity from the predicted capacity is a conservative approach to not over-estimate the capacity as it implies a 95% confidence that the predicted value is not lower than the real value, based on the Gaussian distribution of the prediction error (see Section 5.2.1). The estimation only has to ensure that the the capacity is not over-estimated, thus a one-sided 95% confidence level is sufficiently conservative. This consideration is illustrated in Figure 5.4: It shows the Probability Density Function (PDF) for a normal distribution with $\mu = 40$ and $\sigma = 10$ and the dashed vertical line indicates the point of $\mu - 1.4 \cdot \sigma$. Using $c_i - 1.4 \cdot d_i$ as a bound for the predicted capacity ensures that the real capacity lies within the the blue area and not the red area with a probability of 95%.

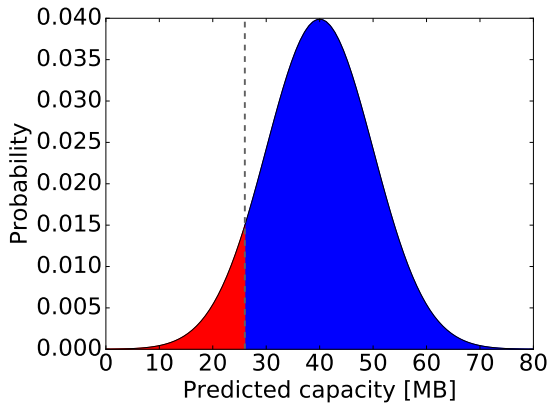


Figure 5.4: Probability density function for data rate prediction

If the prediction horizon is large, the predicted values that lie further in the future become very inaccurate. Also, it is not necessary to download segments for a time slot with low capacity in the far future, if there is a phase with time slots

with sufficient capacity in the near future. Thus the algorithm stops the search for *low* time slots, if there is a time slot in which $c_i - d_i > \max(Q)$. This time slot is called *high*.

The algorithm only downloads a single segment if the conditions are good (line 1). If the data rate in the future decreases for a longer period, the buffer is filled with segments, to avoid playback interruptions in this phase.

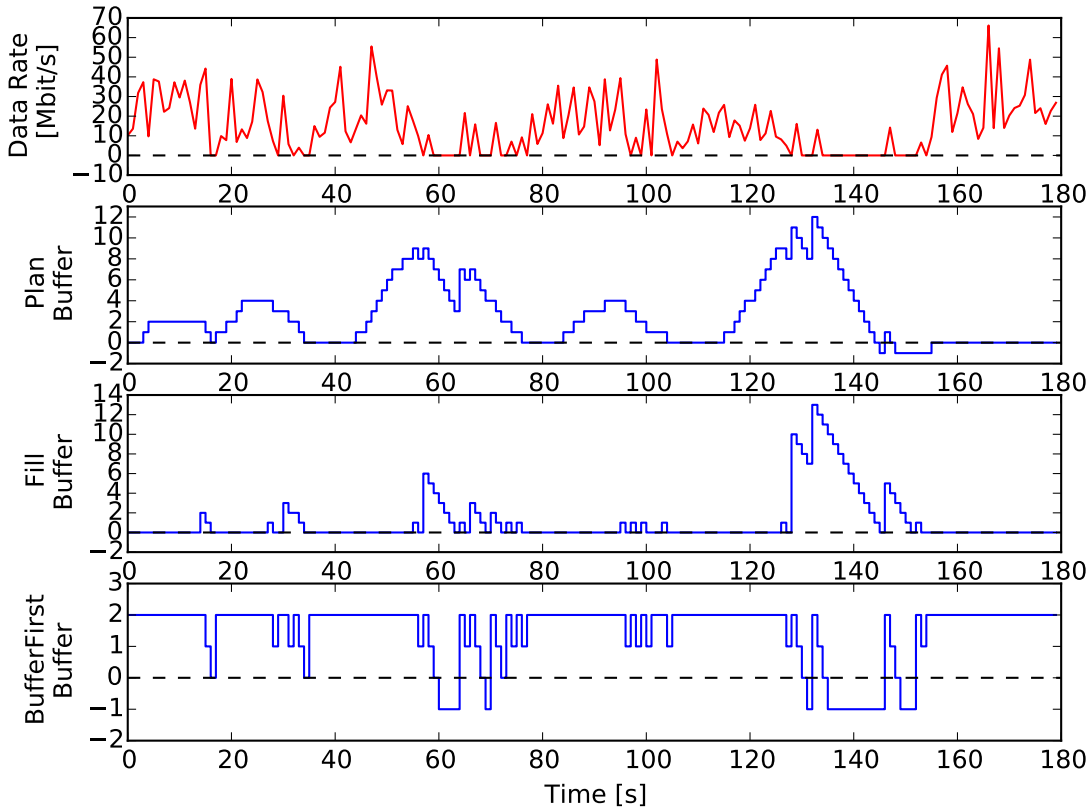


Figure 5.5: Buffering behavior of schedulers

This dynamic adaption of the buffer size for a single user is illustrated in Figure 5.5 in comparison to the buffering behavior of the Fill and BufferFirst algorithms. The figure shows the available data rate for one user and the number of buffered segments for the different schedulers. If the data rate is sufficient, the Plan scheduler and the Fill scheduler only download the next segment and the buffer fill level is zero because the loaded segment is consumed right away. Around seconds 30, 60, 100 and 130 there are phases with no or very low data rate available for the user. While the anticipatory schedulers increase the buffer size before these phases to avoid playback interruptions, the simple greedy algorithm experiences buffer underruns denoted by a buffer fill level of -1. Due to the limited prediction horizon of the Plan scheduler, buffer underruns cannot be prevented completely. If the low data rate phase is too long, like around second 150, not enough segments can be buffered before.

Figure 5.5 illustrates the sliding prediction window of the Plan scheduler. As low data rate time slots enter the window, the algorithm starts to increase the

buffer size, which leads to the pyramidal pattern of the buffer fill level. In contrast to this, the FILL scheduler increases the buffer size shortly before the low data rate phase, which can be seen by the saw tooth pattern in the plot.

Phase 2: Summed Capacity As the second phase, the algorithm has to calculate how much capacity is available until the prediction horizon h, to later determine how many segments should be downloaded and buffered in the current time slot.

The *summed capacity* consists of the current capacity c_c that is known precisely and the sum of the predicted capacities C_{pred} (line 14). The summed capacity is only calculated for time slots up to time h because *Phase 1* only considers segments up to time h. Identical to the calculation in the previous phase we calculate the predicted capacity for future time slot i as

$$c_i - 1.4 \cdot d_i \tag{5.3}$$

to again obtain a 95% confidence that the predicted value is not lower than the real value. The summed capacity is thus calculated as

$$C_s = c_c + \sum_{i=0}^{i \leq h} \max(c_i - 1.4 \cdot d_i, 0) \tag{5.4}$$

Phase 3: Schedule Plan Based on the number of necessary segments from *Phase 1* and the summed capacity from *Phase 2*, the algorithm has to decide which segments to download in which quality level (line 15), considering also how many segments have already been downloaded. This decision, which we call a *plan*, describes what should be downloaded until time $t + h$. This *plan* is then used in the next phase to decide what to actually schedule for time slot t.

To create the *plan*, the algorithm uses the helper function BESTQUALITIES (Algorithm 5.2) which takes the summed capacity C_s, the number of necessary segments n and the set of qualities Q as input. This helper function solves a simplified version of the knapsack problem where the value and weight of an item is equal and from each item there is an unlimited number.

The helper function initially assigns the best quality to each segment (line 3) and then subsequently reduces the quality until the resulting data rate requirement is sufficient (line 8). If the quality for all segments is reduced to the lowest quality and the data rate requirement is not met, the helper function starts to remove segments.

Table 5.2 contains an example of the iterations of the algorithm: The available summed capacity is $C_s = 8$, $n = 3$ segments have to be scheduled and the available qualities are $Q = [5, 3, 2]$. After four iterations, the selected qualities sum up to the available summed capacity, which is a valid quality selection.

Phase 4: Implementation of Plan As the last step, the scheduler checks how much of the *plan* can be scheduled within the current time slot t, by subsequently

Algorithm 5.2 BESTQUALITIES(C_s, n, Q))

```
1: P = q_0, q_1, ..., q_n
2: for all q_i ∈ P do
3:     q_i ← max(Q)
4: end for
5: while Σ_{q_i∈P} q_i > C_s do
6:     if max(q_i ∈ P) > min(Q) then
7:         i_max ← argmax(q_i ∈ P)// if more than one segment has the highest
           quality, choose the one lying in the farthest future
8:         q_{i_max} ← LOWERQUALITY(q_{i_max})
9:     else
10:        REMOVESEGMENT(P)
11:    end if
12: end while
13: return P
```

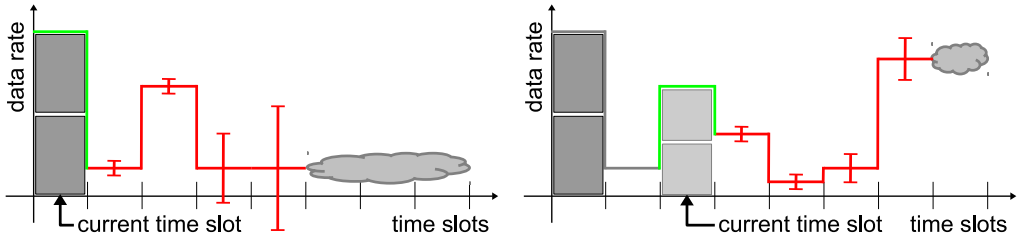
Iteration	$q_i \in P$	$\sum_P q_i$
0	[5, 5, 5]	$\sum = 15 > 8$
1	[3, 5, 5]	$\sum = 13 > 8$
2	[3, 3, 5]	$\sum = 11 > 8$
3	[3, 3, 3]	$\sum = 9 > 8$
4	[2, 3, 3]	$\sum = 8 \geq 8$

Table 5.2: Example of P values in BESTQUALITIES $(C_s = 8, n = 3, Q = [5, 3, 2])$

removing segments from the generated *plan* until the sum of all segments is less than or equal to the current capacity c_c.

Figure 5.6 shows an example of how the algorithm works. In the first time slot (Figure 5.6a), the algorithm detects a *low* phase in the second time slot and thus schedules two segments ($r = 2$) at medium quality. In the second time slot, the available data rate is too low to download any segment. In the third time slot (Figure 5.6b), the algorithm detects a *low* phase in the fifth and sixth time slots, but is only able to schedule two segments at low quality. The algorithm would schedule another segment at low quality in the fourth time slot and incur a playback interruption in the sixth time slot.

When the algorithm is executed in the first time slot, it is aware of a potentially low data rate in the fourth and fifth time slots, but does not include this information into its decision due to two reasons: first, this *low* phase occurs after a *high* phase in the third time slot (see *Phase 1* of the algorithm), and second, the prediction for the fourth and fifth time slots has a high standard deviation and lies in the far future, thus it is included into the data rate calculation with a low weight (see *Phase 2* of the algorithm).



(a) $t = 1$ (b) $t = 3$

Figure 5.6: PLAN algorithm example

5.5 Evaluation

To compare the PLAN scheduler with the perfect prediction schedulers, we first show simulation results without a prediction error ($e = 0$). The simulation parameters are the same as in Section 5.3, except that the number of removed BSs is varied between 0 and 12. All plots are based on multiple simulation runs and show confidence intervals at 95 % confidence level unless they are covered by the plot markers.

5.5.1 Comparison with Perfect Prediction Schedulers

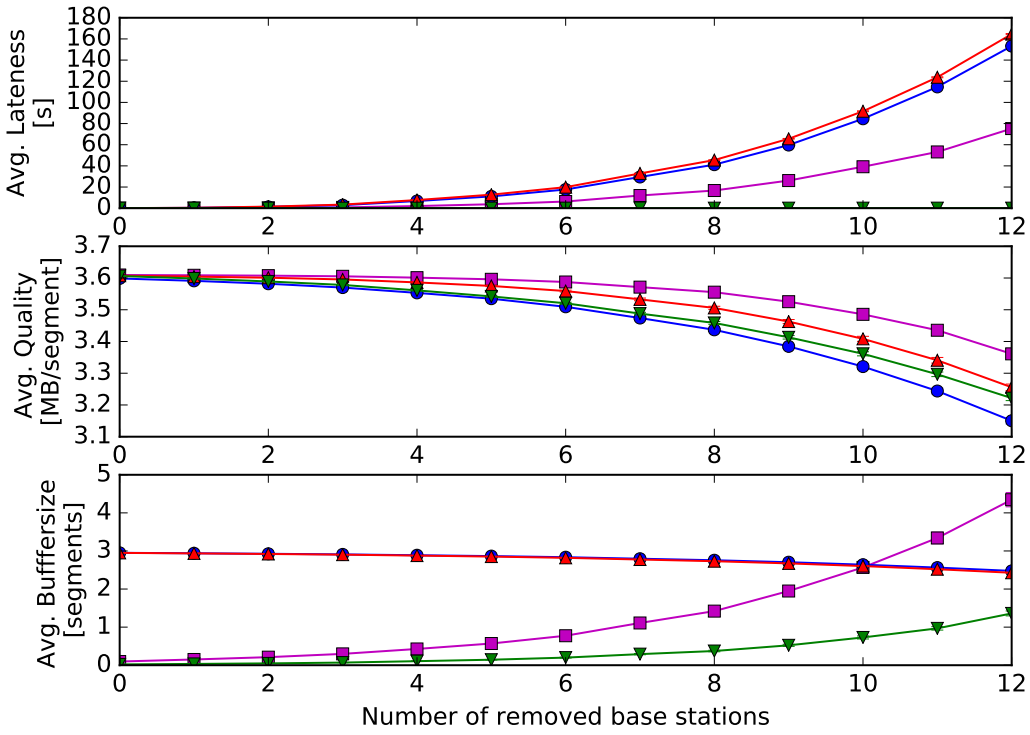
We again evaluate the performance of the schedulers in terms of average lateness (amount of playback interruptions), average video quality, and average buffer level.

Without Prediction Error Figure 5.7 shows the simulation results without any prediction errors ($e = 0$). They are consistent with the previous results in Section 4.5: The greedy schedulers with the fixed buffer size cannot prevent playback interruptions when BSs are removed. The FILL scheduler can prevent playback interruptions even when BSs are removed.

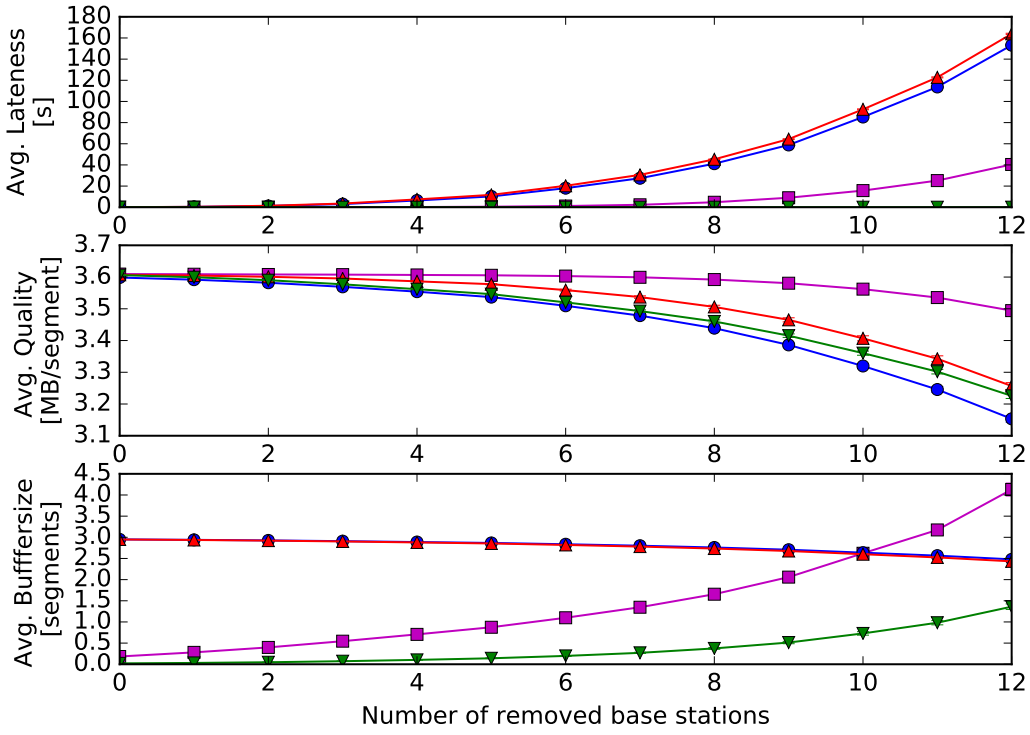
The PLAN scheduler differs from the other schedulers in a very important aspect: When making scheduling decisions at time t, the scheduler only knows the predicted data rates until time $t + k$ because of the limited predicted horizon. In terms of lateness, it performs similar to the FILL scheduler but incurs small playback interruptions when too many BSs are removed. Regarding the average video quality, the PLAN scheduler performs better than the FILL scheduler. Comparing Figures 5.7a and 5.7b shows how the PLAN scheduler benefits from a larger prediction horizon k.

The average buffer level remains almost constant for the greedy schedulers, as expected. For FILL and PLAN, the average buffer level increases as more and more BSs are removed, with PLAN buffering a significantly higher amount of segments.

These results concur with the observations about the buffering behavior in Figure 5.5: FILL buffers segments late (i.e., saw tooth pattern in buffering behavior) at the expense of video quality, while PLAN buffers early (i.e., pyramidal pattern in the buffering behavior) and has more freedom to increase the video quality.



(a) $k = 6$



(b) $k = 12$

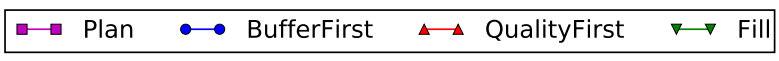
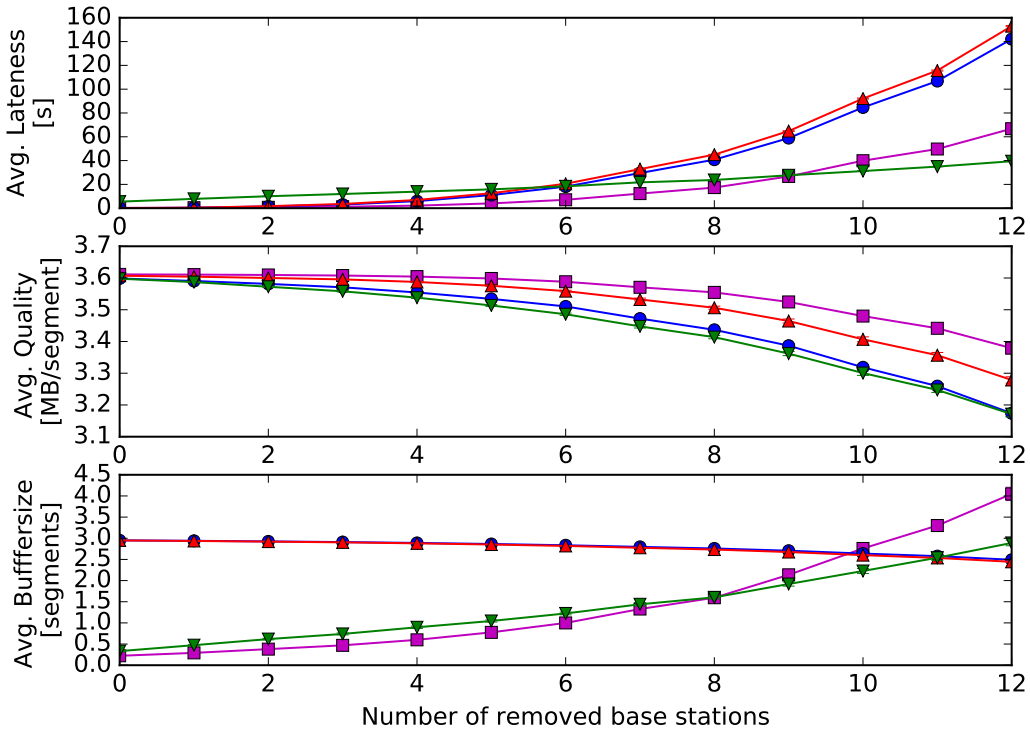


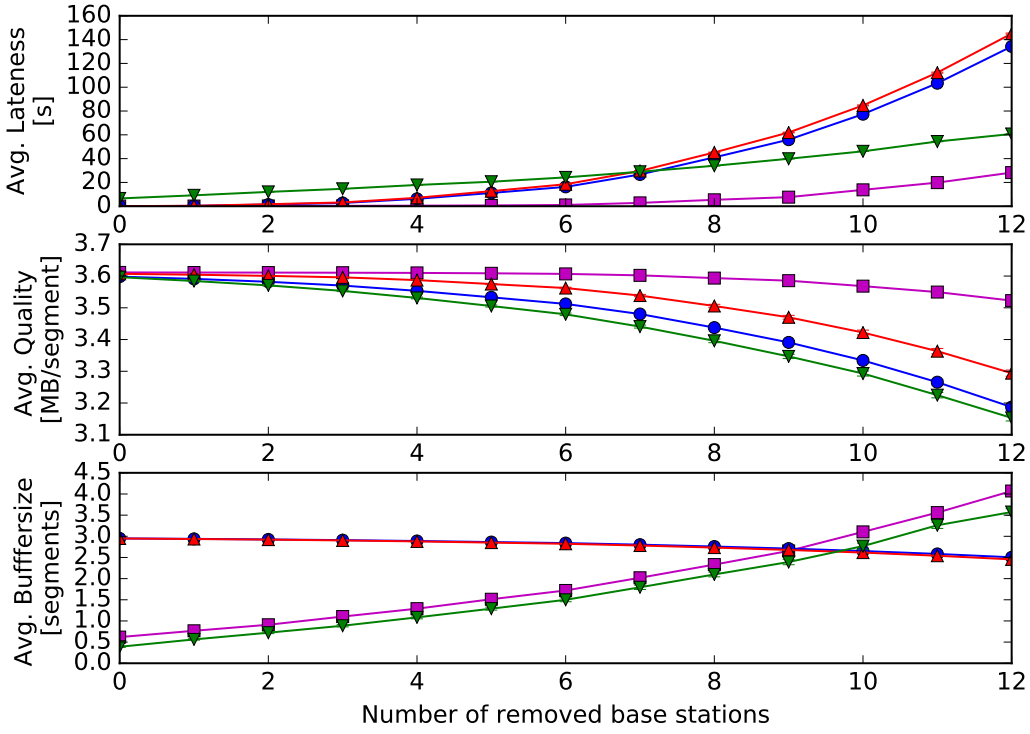
Figure 5.7: Scheduler performance without errors $e = 0$

With Prediction Error Figures 5.8 and 5.9 show the results of simulations where the prediction was erroneous (error factor $e = 2$ and $e = 20$). The prediction horizons k are the same as before. The performance of the greedy schedulers is not affected by the additional error because they do not consider predicted data rates. The FILL scheduler cannot prevent playback interruptions anymore even for small error factors (Figure 5.8). With a large error factor, as shown in Figure 5.9b, the performance of the FILL scheduler becomes as bad as the greedy schedulers.

Because the PLAN scheduler averages out the errors in *Phase 2*, it is not as heavily affected by the prediction errors as the FILL scheduler. For small error factors it performs better than the greedy and FILL schedulers and can keep this level if the error further increases. As soon as the prediction becomes uncertain, the PLAN scheduler outperforms the other schedulers in all points: lower lateness, better quality and more efficient buffering.



(a) $k = 6$



(b) $k = 12$

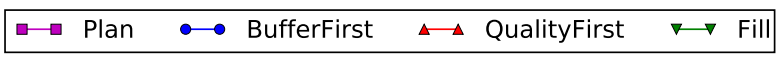
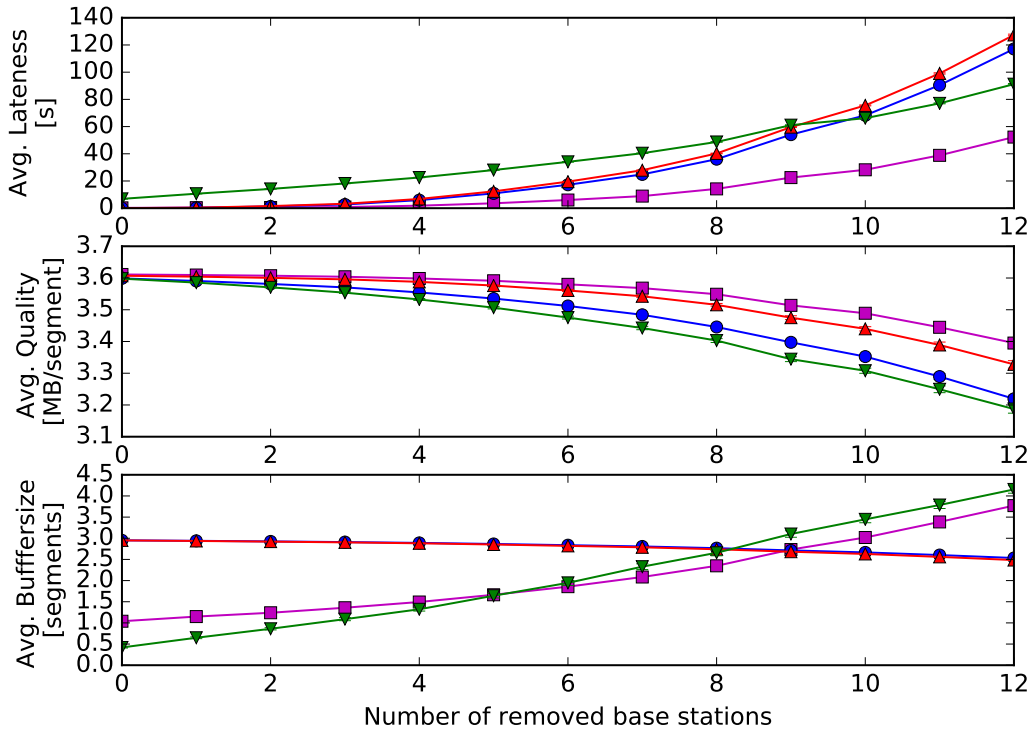
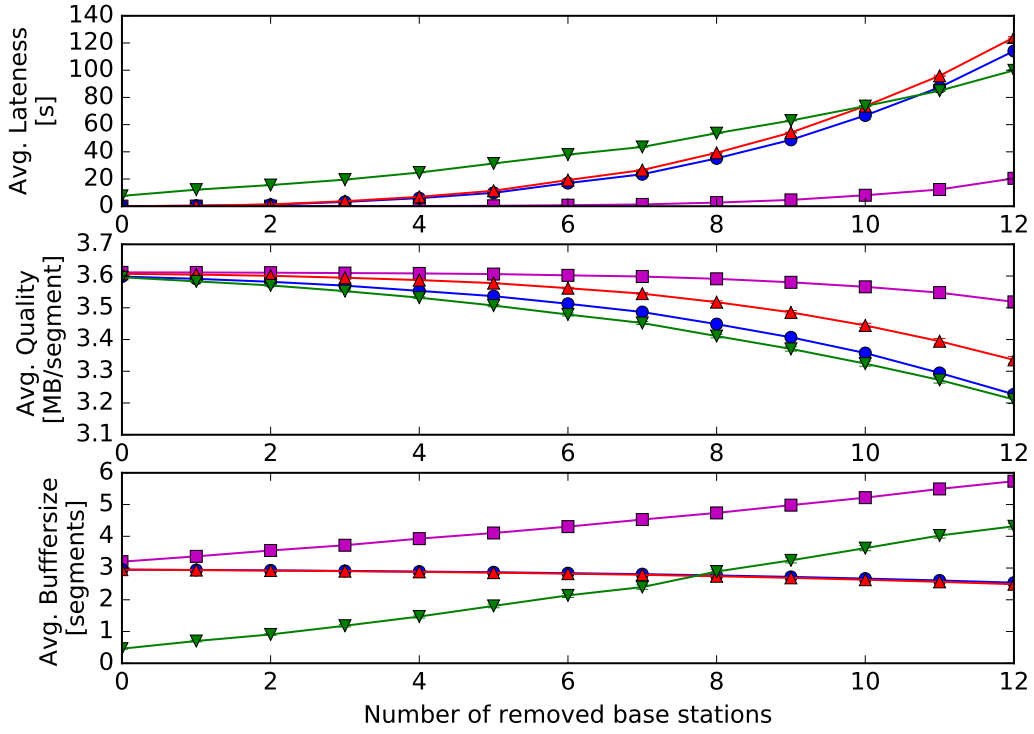


Figure 5.8: Scheduler performance with errors $e = 2$



(a) $k = 6$



(b) $k = 12$

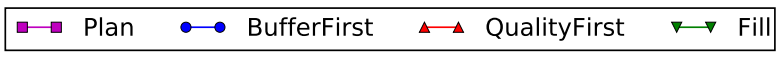


Figure 5.9: Scheduler performance with errors $e = 20$

5.5.2 Influence of the Prediction Horizon

As the PLAN scheduler bases its scheduling decisions only on a limited amount of anticipatory knowledge about future data rates, the size of the prediction horizon plays an important role for its performance. If it is set to $k = 0$, the scheduler only knows the current data rate. If it is set to the duration of the scenario, it knows as much as the other schedulers. But a very high prediction horizon also means that the predicted data rate that lies further in the future would be very inaccurate.

With an increasing number of removed BSs, the gaps between the BSs become larger. If the prediction horizon is too small, the scheduler cannot know about all time slots with low or no available data rate and will not buffer enough segments to avoid a playback interruption. But since in Phase 1 the algorithm stops the search for time slots with low data rate if there is a subsequent phase with high data rate, increasing the prediction horizon to be larger than these gaps has no benefit.

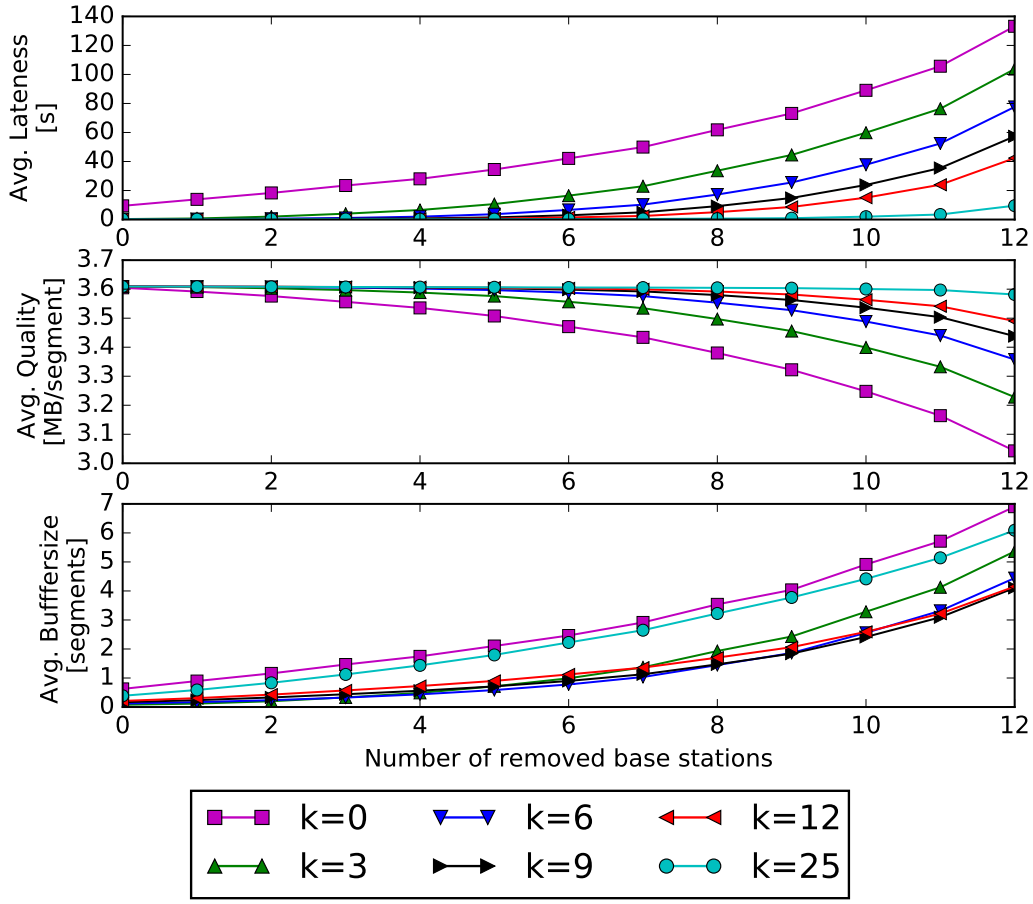


Figure 5.10: Influence of different prediction horizons

Figure 5.10 illustrates these considerations: The prediction horizon should be at least as large as the gaps between base stations. Otherwise playback interruptions occur.

5.6 Summary

In this chapter, I have described how anticipatory download scheduling is feasible with uncertain prediction of available data rates. I have introduced a generic predictor, based on a stochastic model of prediction errors and presented the PLAN algorithm that is able to handle uncertain predictions.

The evaluation results show that the PLAN algorithm is able to significantly reduce playback interruptions compared to greedy scheduling without the prediction of future available data rates. The PLAN algorithm is also able to increase the delivered video quality compared to the FILL scheduler.

These results prove the feasibility of anticipatory download scheduling with uncertain predictions, as they would occur in a real world scenario.

6

Anticipatory Download Scheduling for Energy Efficiency

In the previous two chapters, simulating phases of low available data rate is done by random removal of Base Stations (BSs), resembling users in an area with limited coverage from the mobile access network. The good results in terms of Quality of Experience (QoE) with anticipatory download scheduling in these scenarios led to the question if a similar approach in scenarios with full coverage from the mobile access network can be used to reduce the number of active BSs by switching them off while still maintaining a good QoE for the users. The more BSs would be switched of the less energy the mobile access network would consume.

Together with Philipp Dreimann, I have extended the model of anticipatory download scheduling to also include different power states for BSs. This requires also a new model with a modified assignment of wireless channel resources. Based on the new model, we have extended the Mixed Integer Quadratically Constrained Program (MIQCP) from Section 4.2 and created a new heuristic algorithm. Both

approaches jointly optimize the number of active BSs and the anticipatory download scheduling. We evaluate both approaches in comparison to greedy download scheduling and a simple power cycling mechanism for the BSs.

6.1 Problem Description

Similar to the original problem description of anticipatory download scheduling in Section 4.1, the problem of anticipatory download scheduling with power cycling of BSs can be summarized by two questions:

1. *When* should each segment be downloaded at *which quality* from *which BS*?
2. *Which* BS should be turned on *when*?

To formulate the scheduling problem as an optimization problem we assume a discrete time model. Like in the previous chapters we assume that a video stream is downloaded by all users and streaming starts in the first time slot t_0 and ends in the last time slot $t_{|T|}$. The video is $|T|$ segments long and the length of a video segment is equal to the length of a time slot. Again, this means that for an uninterrupted playback a segment s_i has to be downloaded at the latest in time slot t_i. The quality of the video is determined by the resolution, bit rate, or codec and different qualities result in different file sizes for each segment in each quality. For simplicity we assume that all segments with quality q have the same file size, hence, quality and file sizes are interchangeable. Thus for each segment s, a quality q and a download time t have to be determined per user like in the previous chapters.

The BS power cycling can be modeled in a simple way: For each time slot t a BS a can be switched on or off. If it is switched on, users can be associated with it and the BS can provide data rate to its associated users. For this approach we always assume that there is an always-on macro BS available to provide coverage for voice calls, especially emergency calls, and we only switch on or off additional BSs to provide capacity for data transmissions, i.e., video streaming.

Figure 6.1 illustrates the correlation between the two scheduling decisions and the resulting effects: enabling more BSs potentially increases available data rate, but also increases power consumption. The more data rate is available the more freedom to schedule downloads of segments exists and playback interruptions are reduced. Also, a higher video quality can be downloaded if more data rate is available. Furthermore, additional active BSs generate harmful interference which decreases available data rate. We only model interference as a static, worst-case value according to the GreenTouch model (Section 2.2.2) because computing a dynamic value would introduce too much complexity into the model.

Before introducing our optimization model, we give a short example of how the described scheduling problem works. Figure 6.2a shows how the existing anticipatory scheduling approach from Chapter 4 without power cycling behaves: In time slot 3 there is a decrease in available data rate, so an additional segment has to be downloaded and buffered in time slot 2. This avoids a playback interruption, but additionally available data rate in time slots 1, 5 and 6 is not exploited.

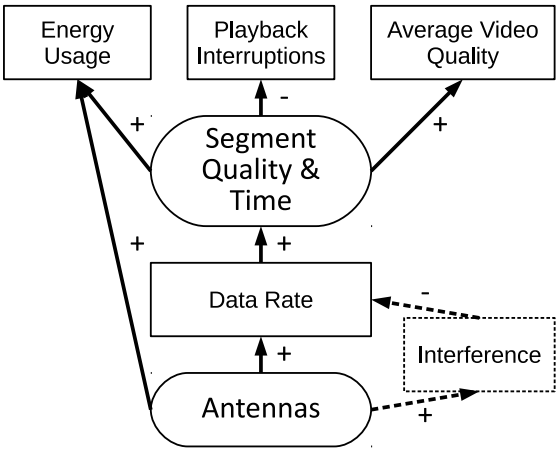
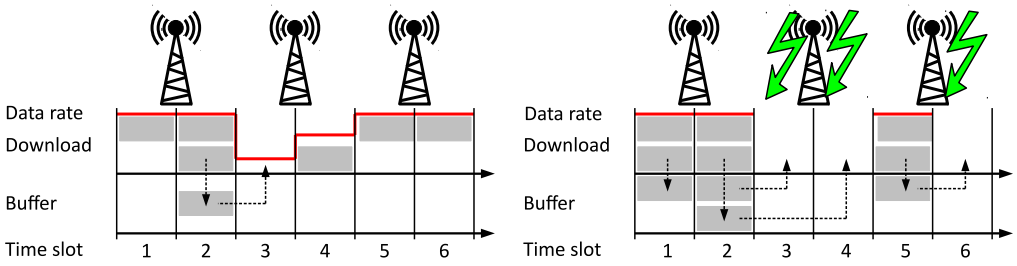


Figure 6.1: Interaction between problem variables and resulting effects



(a) Existing scheduler (download) (b) New scheduler (download + BSs)

Figure 6.2: Example

Our new approach (shown in Figure 6.2b) downloads and buffers additional segments in time slots 1, 2 and 5. With this download schedule, the BSs can be switched off in time slots 3, 4 and 6 without causing any playback interruption. For simplicity, the selection of video quality is omitted in this example.

6.2 Optimal Solution

In this section we describe the optimization problem, which is based on the MIQCP in Section 4.2. Two schedulers, OptBasic and OptFlex, are implemented with this optimization problem. We first describe OptBasic in Section 6.2.1 and then describe OptFlex as an extension of OptBasic in Section 6.2.2.

The input parameters for both optimization problems are defined in Table 6.1 and the decision variables in Table 6.2. The parameters and variables related to the download scheduling are the same as in Chapter 4 and the parameters and variables related to BSs are new in this chapter.

Table 6.1: Input parameters

T	set of time slices, e.g., {0,1,2,3}
S	set of segments to transfer, e.g., {0,1,2,3}
U	set of users, e.g., {0,1,2,3}
Q	set of qualities, e.g., {10, 20, 50}
A	set of BSs, e.g., {0, ... 9}
$D_{u,a,t}$	data rate of user u connected to BS a at time t, e.g., 42
B	resource blocks per BS and time, e.g., {0, ... 99}
$D^B_{u,a,t}$	data rate per resource block of user u connected to BS a at time t, e.g., 0.42

Table 6.2: Variables

$d_{s,t,u} \in \{0,1\}$	deliver segment s at time t to user u
$e_{s,u,q} \in \{0,1\}$	deliver segment s for user u at quality q
$f_{s,u} \in \mathbb{N}$	quality for segment s for user u
$g_{s,t,u} \in \mathbb{N}$	quality for segment s for user u at time t
$l_{s,u} \in \mathbb{N}$	lateness per segment s for user u
$m_{s,u} \in \mathbb{N}$	summed lateness per segment s for user u
$c_{u,a,t} \in \{0,1\}$	connection of user u to BS a at time t
$r_{u,t} \in \mathbb{Q}^+$	data rate per u and time t
$p_{a,t} \in \{0,1\}$	power status of a BS a at time t
$s_{u,a,t,b} \in \{0,1\}$	resource block b assigned to user u at BS a and time t

6.2.1 OptBasic

The most important addition to the optimization problem is the model for power cycling of BSs. Since a scenario consists of multiple BSs, the data rate for each user and BS in each time slot has to be derived. The $D_{u,a,t}$ input parameter contains these data rate values. Data rates are a worst-case estimate with respect to the interference according to the GreenTouch model. First, we compile a list of users that can connect to a BS by checking if the Signal to Interference and Noise Ratio (SINR) is above the minimum threshold of valid SINR values, as defined by the GreenTouch radio model (Section 2.2.2). Each user then gets an equal share of bandwidth per BS to model a simple underlying radio resource scheduler, consistent to the previous evaluations in Chapters 4 and 5. The second step is to use the SINR value to look up the spectral efficiency. This value is then multiplied with the bandwidth to derive the actual data rate for each user per time and BS.

Equation 6.1 defines that a user can be connected to at most one BS per time slot. The available data rate per user and time $r_{u,t}$ is then defined in Equation 6.2.

$$\sum_{a \in A} c_{u,a,t} \leq 1, \forall u \in U, t \in T \tag{6.1}$$

$$r_{u,t} = \sum_{a \in A} D_{u,a,t} \cdot c_{u,a,t}, \forall u \in U, t \in T \tag{6.2}$$

Equation 6.3 ensures that each user downloads each segment exactly once. Equation 6.4 controls that each segment is only downloaded in one quality. Each segment also has to be downloaded once. Equation 6.5 calculates which segment is downloaded in which quality as $f_{s,u}$.

$$\sum_{t \in T} d_{s,t,u} = 1, \forall s \in S, u \in U \tag{6.3}$$

$$\sum_{q \in Q} e_{s,u,q} = 1, \forall s \in S, u \in U \tag{6.4}$$

$$f_{s,u} = \sum_{q \in Q} e_{s,u,q} \cdot q, \forall s \in S, u \in U \tag{6.5}$$

The time when a segment with a certain quality is downloaded is defined in Equation 6.6. A segment can only be downloaded if there is enough available data rate as defined in Equation 6.7.

$$g_{s,t,u} = f_{s,u} \cdot d_{s,t,u}, \forall s \in S, t \in T, u \in U \tag{6.6}$$

$$\sum_{s \in S} g_{s,t,u} \leq r_{u,t}, \forall u \in U, t \in T \tag{6.7}$$

Equation 6.8 defines the per user and per segment lateness $l_{s,u}$. The individual lateness values per user and segment are summed up as $m_{s,u}$ in Equation 6.9.

$$l_{s,u} = \sum_{t \in [s+1, \max(T)]} d_{s,t,u} \cdot t, \forall s \in S, u \in U \tag{6.8}$$

$$m_{s,u} = \sum_{x \in [0, s+1]} l_{x,u}, \forall s \in S, u \in U \tag{6.9}$$

So far, all BSs are modeled as always powered on. Thus we need a new state variable $p_{a,t}$ per BS and time slot to model whether a BS is enabled. This variable is used to determine to which BSs users can connect. Equation 6.10 restricts connections only to enabled BSs (variable $c_{u,a,t}$).

$$c_{u,a,t} \leq p_{a,t}, \forall u \in U, a \in A, t \in T \tag{6.10}$$

The inverse direction needs to be considered too: Equation 6.11 makes sure that a BS is disabled if no users are connected.

$$p_{a,t} \leq \sum_{u \in U} c_{u,a,t}, \forall a \in A, t \in T \tag{6.11}$$

Additionally, users only need to be connected to a BS in a time slot if they download segments in that time slot. The sum of segments has to be rounded up

in case segment qualities smaller than 1 are available. This constraint is shown in Equation 6.12.

$$c_{u,a,t} \leq \lceil \sum_{s \in S} g_{s,t,u} \rceil, \forall u \in U, a \in A, t \in T \tag{6.12}$$

OptBasic combines the different goals video quality, lateness, and enabled BSs in an objective function with weight factors, similar to the objective function in Section 4.2. For the objective function in Equation 6.13, each goal has an individual weight factor to create a trade-off between the different goals: W_q for video quality, W_l for lateness and W_p for enabled BSs.

$$\begin{aligned} \text{maximize: } & W_q \cdot \sum_{s \in S, u \in U} f_{s,u} \\ & - W_l \cdot \sum_{s \in S, u \in U} m_{s,u} \\ & - W_p \cdot \sum_{a \in A, t \in T} p_{a,t} \end{aligned} \tag{6.13}$$

OptBasic already takes a long time to solve (see Section 6.4.2), limiting the size of evaluation scenarios. This predicament notwithstanding, we wanted to investigate if a more flexible and realistic model for the assignment of data rates to users would show any significant gains. Thus we extended the model for the OptFlex scheduler.

6.2.2 OptFlex

To model the wireless channel resources more realistically we use Resoure Blocks (RBs). RBs are generic resource allocation units for the bandwidth of a BS to a user, similar to Physical Resource Blocks (PRBs) in LTE.

Because of the RBs, we need a new input parameter for the data rate $D^B_{u,a,t}$ which has to be used differently than $D_{u,a,t}$. A new variable $s_{u,a,t,r}$ is used to model the assignment of RBs to users. Instead of dividing the bandwidth by the number of users, the data rate is calculated using several intermediate steps. First, we assume each user is the only user connected to a BS, this results in a data rate per user hypothetically using the cell alone. Second, this hypothetic data rate has to be shared with other users in the cell. For this the hypothetic data rate is divided by the total number of RBs ($|B|$). The result is a data rate $D^B_{u,a,t}$ per user and per RB which can multiplied with the number of assigned RBs of the user in the linear equations as follows. Again, these data rates are a worst-case estimate with respect to the interference according to the GreenTouch model. The number of RBs per BS and time slot is limited by Equation 6.14.

$$\sum_{u \in U, b \in B} s_{u,a,t,b} \leq |B|, \forall a \in A, t \in T \tag{6.14}$$

Equation 6.15 and Equation 6.16 control from which BS a user can use an RB. To use an RB, a user must be connected to an enabled BS (Equation 6.15). If a

user has no RBs assigned, it should not be connected (Equation 6.16).

$$s_{u,a,t,b} \leq c_{u,a,t}, \forall u \in U, a \in A, t \in T, b \in B \tag{6.15}$$

$$\sum_{b \in B} s_{u,a,t,b} \cdot D^B_{u,a,t} \geq c_{u,a,t}, \forall u \in U, a \in A, t \in T \tag{6.16}$$

The data rate per user and time slot can now be computed with Equation 6.17. This replaces Equation 6.2, the definition of a data rate per user and time as used by OptBasic.

$$dr_{u,t} = \sum_{a \in A, b \in B} D^B_{u,a,t} \cdot s_{u,a,t,b}, \forall u \in U, t \in T \tag{6.17}$$

Because of the large number of new variables in this version of the optimization problem, it takes two orders of magnitude longer to solve it compared to OptBasic (see Section 6.4.2).

6.3 Two-Phase Algorithm

Because of the complexity and long solving time of the optimization problem, we have implemented a heuristic scheduling algorithm called 2-Phase. It operates in two phases: The first phase finds the highest quality per segment that can be scheduled without incurring lateness assuming all BSs are enabled. Based on this schedule, all BSs that are not needed for downloads are disabled in the second phase. The results from the two phases are then used to determine both the download schedule for each user (*when* should each segment be downloaded at *which quality*?) and the power cycling schedule (*which* base station or BS should be turned on *when*?).

The initial goal of the scheduler is to find a segment quality assignment that does not introduce lateness. The power consumption is only considered afterwards and reduced as much as possible. We now describe the two phases of 2-Phase in Sections 6.3.1 and 6.3.2. The overall structure of the algorithm is illustrated in Figure 6.3.

6.3.1 Quality selection phase

Because in this phase of the algorithm all BSs are still switched on and are able to provide capacity to the users to download segments, we can use a simple algorithm to select segment qualities that is simpler than for example the Fill algorithm from Section 4.3.

The first step of this phase is to calculate the anticipated available data rate for each user in each time slot with each BS $D_{u,a,t}$.Then, the algorithm executes Algorithm 6.1 for each user. The goal of the algorithm is to find the highest video quality that can be downloaded by a user at the anticipated data rate. The algorithm starts without an initial solution Z (line 2). A list of all possible quality assignments is stored in $Q_{options}$ in line 3. The quality assignments are sorted increasing from the last to the first segment. In line 4, variable S is set

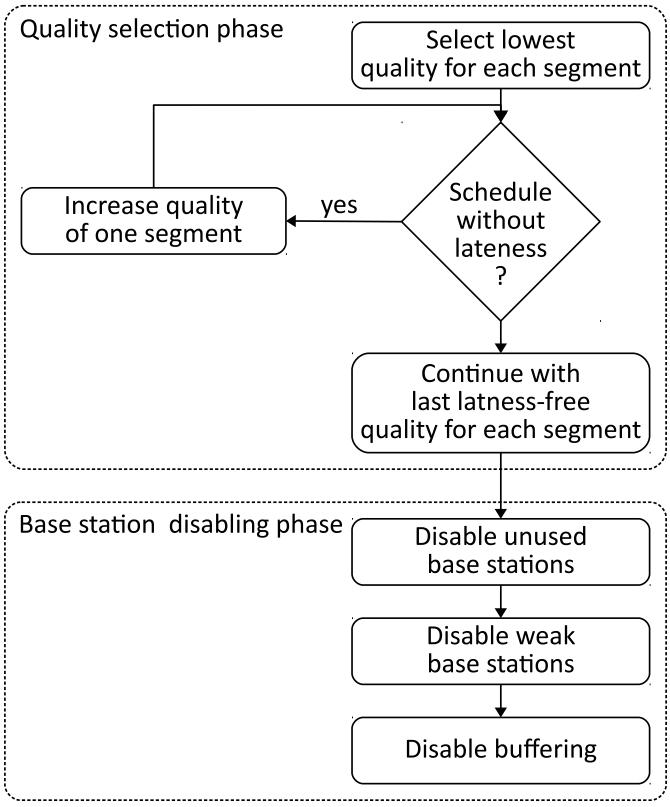


Figure 6.3: Flowchart for 2-Phase

to a list in which each value represents the number of segments that have to be downloaded to support the lowest quality. These two variables are the basis for the following functions.

The crucial problem of this algorithm is to determine in which quality each segment can be downloaded while keeping the lateness low and the average quality high. While determining the quality, it also generates a list of BSs for each time slot which provide sufficient data rate for the segment downloads. The algorithms solves the quality assignment problem by checking all possible quality assignments, starting from the lowest to the highest using a fast validation loop over all options (lines 8 to 16). For example, the content of the $Q_{options}$ in line 3 for a scenario with $Q = \{1, 2\}$ and $|T| = 2$ is $\{\{1, 1\}, \{1, 2\}, \{2, 1\}, \{2, 2\}\}$, generated by qualityOptionsIncr$(Q, |T|)$.

The semantics of the result of the segmentDemand function are explained with the example in Figure 6.4 and the implementation of the function is shown in Algorithm 6.2. The parameters of the example scenario are $|T| = 4$ and $Q = \{1\}$. In this example, only one BS is available. Available data rate per time slot is given by the red line. The blue boxes 0–3 represent the data rate demand of the lowest quality q_{min} in the algorithm (line 2).

$S_{|T|}$ of the last time interval is initialized with the duration $|T|$ of the scenario (line 3). This means, if all segments (of which there are $|T|$ many) are downloaded, the schedule has no playback interruptions. For time slot $t = 3$, there is enough

Algorithm 6.1 QUALITYASSIGMENT$(u, D_{u,a,t})$

```
1:  // qualities Q and duration |T| from scenario
2:  Z ← ∅ // no initial solution
3:  Q_options ← QUALITYOPTIONSINCR(Q, |T|)
4:  S ← SEGMENTDEMAND(u, r_{u,a,t})
5:  for all q ∈ Q_options do
6:     A ← ∅ // used BSs
7:     c ← 0 // count of downloaded segments
8:     for all t ∈ T do // iterate over all time slots
9:        a ← SUFFICIENTBSS(t, c, q, S)
10:       if |a| ≠ 0 then
11:          c ← c + MINDOWN(t, c, q, a)
12:          A ← A ∪ {a}
13:       else
14:          return Z
15:       end if
16:    end for
17:    Z ← (q, A)
18: end for
```

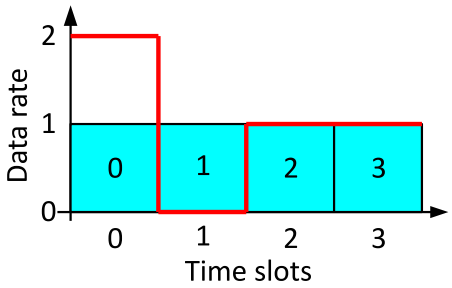


Figure 6.4: Segment quality and available data rate example

data rate available to download segment three. Therefore $D_3 = 3$ (line 16) and segment 3 is added and then removed from the list of late segments L (lines 6 and 15). In time slot $t = 2$, the situation is the same as in time slot $t = 3$. Sufficient data rate is available, thus $D_2 = 2$. In time slot $t = 1$, the data rate is not even sufficient to download the lowest quality. If the segment cannot be downloaded in an earlier time slot, the lateness would increase. This information is stored by setting $S_0 = 2$ and $S_1 = 2$. Additionally, segment 1 is kept in the list of late segments, due to $i = 0$ and $C = \emptyset$. At time slot $t = 0$ sufficient data rate for two segments is available and segments 1 and 2 are in the list of late segments L. Therefore $c = 2$ and $C = \{1, 2\}$ after the while loop, and both segments are removed from the list of late segments L. S_{-1} is set to 0 which indicates that there is a schedule for this scenario that incurs no lateness.

In the validation loop in Algorithm 6.1 (lines 5 to 18), the previously computed possible quality assignments are validated. First, the list of usable BSs and the counter of downloaded segments are initialized (line 6 and 7). Then, for all time slots, it is checked whether there are BSs available that provide sufficient data

Algorithm 6.2 segmentDemand($u, D_{u,a,t}$)

```
1: // qualities Q, duration |T| and BSs A from scenario
2: q_min ← min(Q)
3: S_|T| ← |T|
4: L ← ∅ // late segments
5: for all t ∈ reversed(T) do
6:    L ← L ∪ {t}
7:    r ← max_{a∈A}(D_{u,a,t})
8:    i ← 0
9:    C ← ∅ // feasible segments
10:   while (r ≥ q_min)and(c ≤ |L|) do
11:      r ← r − q_min
12:      C ← C ∪ {L_i}
13:      i ← i + 1
14:   end while
15:   L ← L \ C
16:   S_{t−1} ← S_t − i
17: end for
18: return S
```

rate. The invariant is that c is always greater or equal to S_t, which is checked in line 11.

sufficientBSs() schedules as many segments from q as permitted by the available data rate of the BSs. If this number of newly downloaded segments added to c of previously downloaded segments is greater or equal to S_t, then the BS is returned. Additionally, $\emptyset$ is added to the list of BSs if c is greater or equal to S_t without new downloaded segments. If sufficientBSs() does not return any BSs then there is no solution for this quality assignment and the given BSs. This validation approach does not find all possible BS combinations for a given schedule. This is due to the fact that c is only increased by the minimum number of segments that all sufficient BSs can download. As soon as a quality assignment is not valid, the last successfully tested solution is returned together with the list of BSs per time (line 14).

At the end of this phase the returned values from qualityAssignment() (Algorithm 6.1) are passed on to the next phase as two sets of variables $q_{u,s}$ and $A_{u,t}$, where $q_{u,s}$ is the best valid quality assignment per user u for segment s and $A_{u,t}$ the BSs which can fulfill the quality assignment $q_{u,s}$ for user u and time t.

6.3.2 Base station disabling phase

The second phase of the algorithm tries to find and disable unneeded BSs. The previous phase only decided in which quality which segment can be downloaded and from which potential BSs in each time slot. An assignment of users per time slot to an exact BS is needed for a valid schedule. BSs per time slot can be disabled until as few BSs as possible are left. 2-Phase sequentially uses three strategies to disable BSs.

Unused base stations The first strategy is to search for completely unused BSs and to disable them. The algorithm iterates over all possible time slots and checks per time slot for each BS if any user can use it according to $A_{u,t}$. If not, the BS is added to the list of disabled BSs for that time slot.

Weak base stations The second strategy removes those BSs that provide less data rate than others. This is achieved by creating a list of all users from $A_{u,t}$ that can connect to an BS per time slot. The data rates of those users are then summed per BS and sorted in increasing order. The algorithm disables BSs for that time slot until one BS remains usable per user.

Segment buffering The third strategy is to disable BSs if their data rate is not essential. Not essential means that the data rate to download segments in a time slot t is also available in the previous time slot $t-1$. Thus the segments are scheduled for the previous time slot $t-1$ and the BS can be disabled in time slot t.

6.4 Evaluation

Because of the complexity of the optimization problem, the evaluation is twofold: First, we compare the optimization problem, the heuristic algorithm and the algorithms from Chapter 4 in a small scenario. Second, we compare the heuristic algorithm with the QualityFirst greedy algorithm (Section 4.4).

6.4.1 Scenarios

Both evaluations use the radio and power models from the GreenTouch project (Section 2.2.2). Segment sizes are again set to 1.413 MB for low, 2.951 MB for medium, and 3.613 MB for high video quality.

The first scenario (Three BSs) consists of three base stations placed in a line and a variable number of stationary users uniformly placed between them. We use this scenario once with macro BSs (one undirected sector, 8x2 MIMO, rural environment, 80% of rural ISD: 3464 m) and once with pico BSs (ISD 200 m) according to the GreenTouch model. Both models for BSs have different power consumption characteristics and are thus interesting for this evaluation.

For the second scenario (Train) we simulate a train ride on a regional train: A group of users moves along a line with several stations in between where the train stops. Based on a real train schedule connecting the cities Paderborn and Herford[1] in Germany, the train moves for a certain amount of time from one station to the next station and then stops there for one or two minutes, as listed in Table 6.3. Each station is equipped with a pico base station that can be turned on to allow the users on the train to download and buffer more video segments.

[1] Kursbuchstreckennummer 405, see `http://kursbuch.bahn.de`, accessed 10.03.2015

Table 6.3: Train ride Paderborn-Herford, all times are given in minutes.

Station	Journey	Stop
Paderborn Hbf	-	2
Altenbeken	12	1
Sandebeck	11	1
Leopoldstal	3	1
Horn-Bad Meinberg	2	2
Detmold	8	2
Lage (Lippe)	6	2
Sylbach	3	1
Schoetmar	3	1
Bad Salzuflen	2	1
Herford	6	2

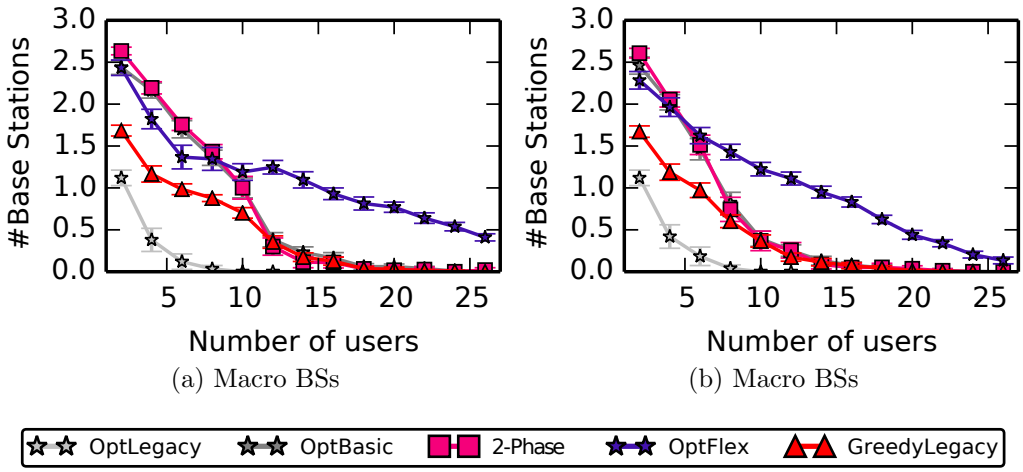
We compare five different schedulers in this evaluation:

- OptBasic is the optimization problem without flexible data rate assignment (Section 6.2.1)
- OptFlex is the optimization problem with flexible data rate assignment (Section 6.2.2)
- 2-Phase is the heuristic scheduler (Section 6.3)
- OptLegacy is the existing optimization problem from Chapter 4 without explicit power cycling. BSs are initially all switched on and only BSs with no associated users are considered switched off in a post processing step.
- GreedyLegacy is the QualityFirst greedy scheduler (Section 4.4) together with BS power cycling in a post processing step.

All plots show confidence intervals at 95% confidence level unless they are covered by the plot markers.

6.4.2 Three BSs Scenario Results

The most significant result in this scenario is the number of disabled BSs as shown in Figures 6.5a and 6.5b. In both the macro and pico scenarios OptFlex can disable the overall highest number of base stations because it can schedule with the highest degree of freedom by being able to flexibly assign data rates to users. It is closely followed by OptBasic and 2-Phase which perform equally. With more than 15 users neither scheduler is able to disable any BS as the load is too high. With GreedyLegacy a few BSs can be disabled for a small number of users because this scheduler buffers segments and does not have to download a new segment in each time slot. OptLegacy performs worst as it tries to minimize



(a) Macro BSs (b) Macro BSs

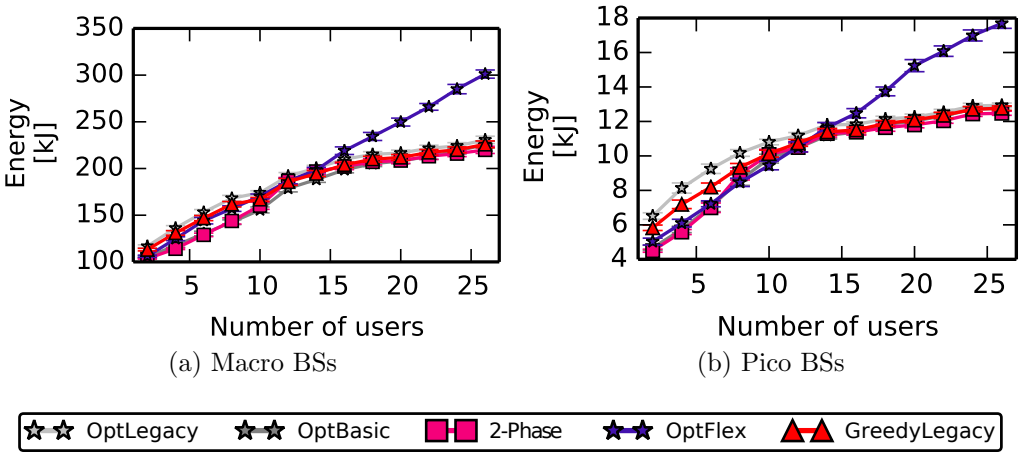
Figure 6.5: Three-BSs scenario: disabled BSs

buffering of segments and downloads segments in almost all time slots without considering energy efficiency.

As the power consumption of the mobile access network is not only influenced by the enabled BSs but also depends on the traffic load, the overall energy usage is different from the number of enabled BSs, as shown in Figures 6.6a and 6.6b. For a low number of users OptBasic and 2-Phase achieve a slightly lower energy usage than the other schedulers. For a higher number of users OptFlex uses significantly more energy. This is because this scheduler schedules more data to be downloaded as it uses a higher video quality (Figures 6.7a and 6.7b). Considering both energy usage and video quality together, the energy used per transferred bit is similar for all schedulers.

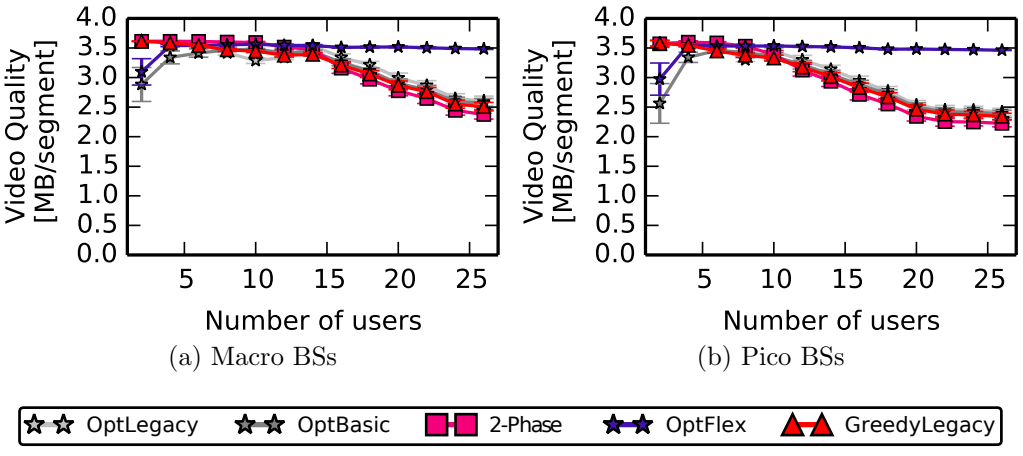
The overall energy usage if no power cycling scheme is employed is between 113.4 kJ (no load) and 399 kJ (full load) for macro BSs and between 7.2 kJ (no load) and 19.8 kJ (full load) for pico BSs. Thus all presented power cycling schemes reduce the energy usage.

Figure 6.8 shows the average running times for the different scheduling algorithms in the pico BSs case. The heuristic algorithms are around two orders of magnitude faster than the simple optimization problems OptBasic and OptLegacy. The more detailed optimization problem OptFlex takes around two orders of magnitude longer to solve than the simple problems.



(a) Macro BSs (b) Pico BSs

Figure 6.6: Three-BSs scenario: energy consumption



(a) Macro BSs (b) Pico BSs

Figure 6.7: Three-BSs scenario: average video quality

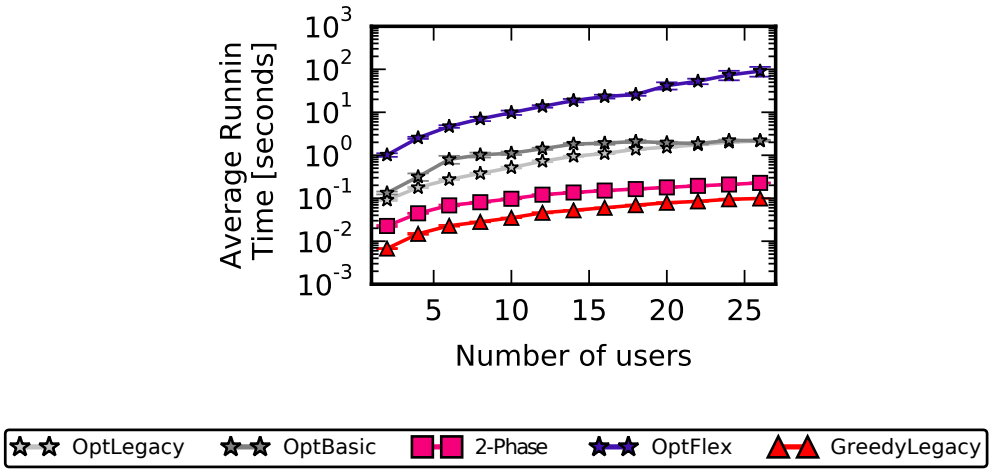
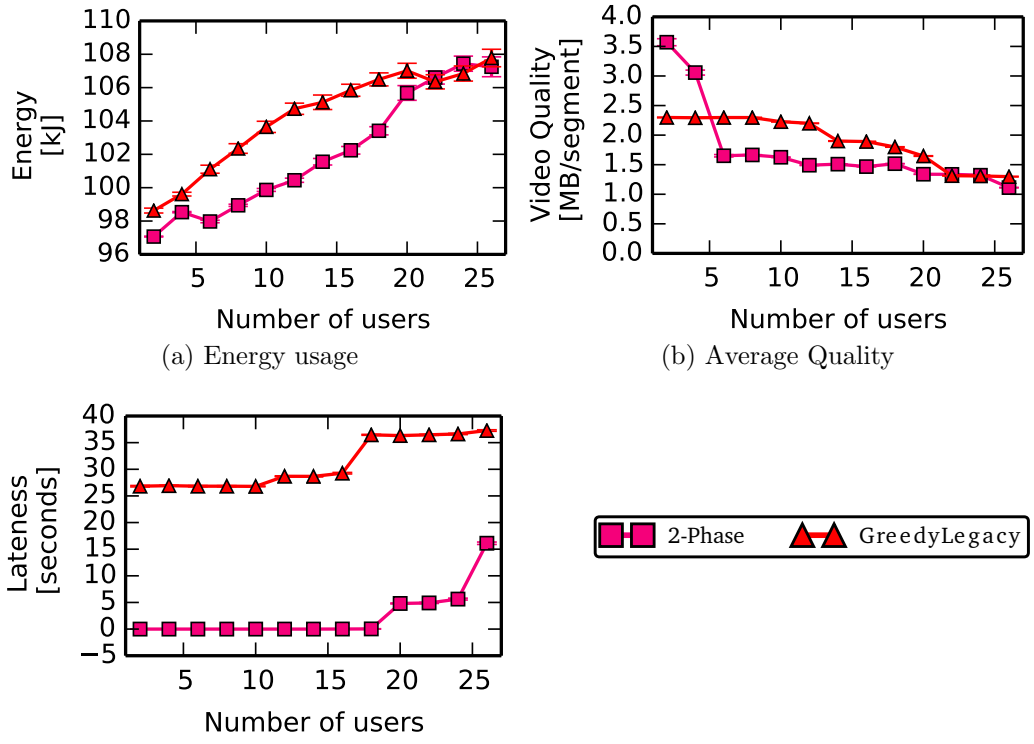


Figure 6.8: Three-BSs scenario: average running time

6.4.3 Train Scenario Results

As the train scenario is too big to solve with the optimization-based schedulers, we only compare 2-Phase and GreedyLegacy. For up to 20 users 2-Phase uses significantly less energy (Figure 6.9a). For a very low number of users 2-Phase delivers a higher video quality, but for more users it drops slightly below the quality delivered by GreedyLegacy (Figure 6.9b). This drop in video quality is mitigated by the significantly lower occurrence of video interruptions (lateness) with 2-Phase (Figure 6.9c).

These results show that our algorithm also works in a practical scenario and is able to reduce energy usage by maintaining a satisfying QoE for the users. The overall energy consumption without any power cycling scheme is between 190.08 kJ (no load) and 522.72 kJ (full load). Thus both power cycling schemes reduce the energy usage.



(a) Energy usage

(b) Average Quality

(c) Average Lateness

Figure 6.9: Train scenario results

6.5 Summary

In this chapter, I have described how the approach for anticipatory download scheduling can be used to reduce the energy consumption of a mobile access network. I have shown an optimization problem as well as an heuristic algorithm that incorporate the aspect of BSs power cycling.

The evaluation results indicate a significant reduction of energy consumption compared to using no power cycling scheme and a smaller, but still significant reduction comparing a simple power cycling scheme with greedy download scheduling to the integrated power cycling scheme with anticipatory download scheduling.

7

Anticipatory Download Scheduling Prototype

Although the simulations in the previous chapters are designed to relate to real-world scenarios, they do not answer the question how anticipatory download scheduling can be integrated into a real video streaming application. In this chapter, I present how anticipatory download scheduling can be integrated into a mobile access network and how the HTTP Live Streaming (HLS) protocol can be extended to include anticipatory download scheduling. I have used this protocol extension to set up a testbed and evaluate the scheduling algorithms from Chapter 4. The testbed has been set up in the context of the Smarter Phones And Networks (SPAN) project together with Johannes Blobel, Philipp Dreimann, Christoph Schniedermeier und Stefan Valentin.

7.1 System Design

In this section I describe how anticipatory download scheduling can be integrated into a real system, using existing tools and existing protocols with backwards-compatible extensions. I first explain the design decisions and their implications on the system behavior and then continue with the system architecture and its interfaces in Section 7.1.2.

7.1.1 Design Decisions

The implementation of anticipatory download scheduling requires changes to existing systems of the mobile access network and the video content provider, to control when which segments of a video are downloaded by User Equipments (UEs). In order to implement these changes, two design decisions with different advantages and disadvantages/costs have to be made:

- Should the buffering behavior be controlled at the UE or in the network?
- Should arbitrary or only pre-selected content providers be supported?

These design decisions have direct implications on the buffering behavior of the system regarding buffering and playback interruptions. Every combination of the design decisions results in the following requirements and capabilities of the system:

1. *No download control at UEs and arbitrary content providers*
   This implementation requires Deep Packet Inspection (DPI) on the network to separate video traffic from other traffic or video traffic from supported video content providers and video traffic from unsupported video content providers, as long as there is no list of preselected, supported video content providers. That imposes additional cost and requires additional processing for the network operator. When there is no modified UE which allows to control the buffer, I assume that the UE downloads greedily. To implement the schedule I can only control the data flows in the network. I can prevent excessive buffering only by limiting the connection speed for a UE and therefore prevent the UE from downloading more segments than it should. But since I cannot force a UE to buffer more than it wants to (greedy behavior), playback interruptions cannot be avoided.

2. *No download control at UEs and pre-selected content providers*
   When implementing the buffer control mechanisms only for pre-selected content providers which I know beforehand, the separation of video traffic from other traffic becomes trivial, in contrast to the previous case. The problem with playback interruptions, however, still remains the same.

3. *Download control at UEs and arbitrary content providers*
   When I have the means to control the download behavior of a UE, e.g., by a modified version of the video player, I can prevent excessive buffering and minimize playback interruptions by explicitly instructing the UE from the scheduler how many segments it should download at a certain time. A modified software could also support the separation of video traffic from other traffic and the identification of the video content provider, e.g., by sending all video requests over a special proxy.

4. *Download control at UEs and preselected content providers*
   If I can fully control the download behavior and can easily separate video traffic from other traffic the implementation of the system becomes most

easy. I then can optimize the buffer sizes on the UEs with little additional complexity on the network side.

Although implementing my approach with the maximum level of control on both the UEs and the content providers is the easiest way, a trade-off has to be accepted here: Limiting the available content providers to a selected few also limits the usefulness to the users. However, the best performance can only be achieved with modifications to the UE, otherwise there is no means to reliably prevent playback interruptions and preventing excessive buffering is difficult to implement.

7.1.2 Architecture and Implementation

For my prototype implementation I chose to use a modified video player on the UEs. With this setup I can fully control the download behavior and I can analyze the performance of the system. My implementation supports arbitrary content providers but in my evaluation I use my own video source to eliminate external influences.

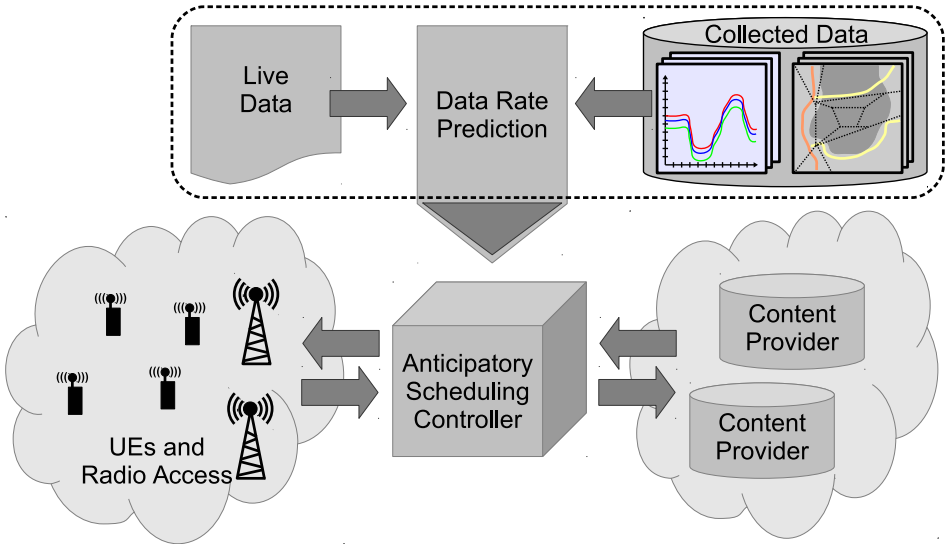


Figure 7.1: Architecture

To implement my schedulers I assume an overall architecture as depicted in Figure 7.1. This architecture does not require any changes to current mobile access networks and can be implemented in a cellular network as well as in a wireless LAN scenario since the scheduler is implemented in higher layers. It also does not require any changes to the content provider since all scheduling decisions and the schedule is enforced in the *Anticipatory Scheduling Controller*. The *Data Rate Prediction* works based on live data, e.g. GPS data, and previously collected data, e.g. coverage maps or traces, depending on the prediction techniques it uses. Predicted data rates are then provided to the *Anticipatory Scheduling Controller* as is the central entity in this architecture. It intercepts the requests from the UEs

to the content providers. It can then perform the download control and quality selection with the following three steps:

1. Intercept the video request from the UE and analyze it (video data rates, available HLS variants)
2. Calculate a schedule based on video data and predicted information on future data rates
3. Control the buffering behavior of the UE according to the schedule

To do so, the *Anticipatory Scheduling Controller* could be configured as an HTTP proxy as HLS video requests are transported via HTTP. This can be enforced in mobile access networks by the network operator or done voluntarily by the users. Both operators and users have incentives to do so: less load on the network, better QoE for the users.

7.2 Prototype Implementation

In this section I first explain the implementation details and adjustments to the HLS protocol necessary to use the scheduling algorithms in Section 7.2.1. The concrete Testbed implementation which I used to verify my simulation results is described afterwards in Section 7.2.2.

7.2.1 Protocol Extension

My protocol extension is based on the HLS protocol, which I introduced in Section 2.4.1.

To control the buffering behavior of HLS players, I need a method to pass messages to them. HLS players have no interface to receive control data besides playlists and segments via their own `HTTP-GET` requests. I intercept the requests for playlists and modify the replies in the anticipatory scheduling controller.

The controller is aware of the schedule but also needs a means of inserting buffering instructions in the playlists. Thus, I introduce two new tags to HLS playlists: `BUFFERSIZE` and `REFRESH`. Both are defined as natural numbers including 0. These new tags are backwards-compatible because the HLS standard instructs players to ignore tags which they do not recognize [PMA13].

`BUFFERSIZE` sets the size of the HLS player buffer to the given value. Up to this amount of segments, the player will greedily try to download more segments. If there are more segments in the buffer than instructed, the buffer content is played and no downloaded segments are discarded. As soon as there are fewer segments in the buffer than the given limit, the HLS player downloads additional segments to fill the buffer.

The `REFRESH` parameter instructs the HLS player to refresh the playlists every `REFRESH` seconds. This refresh will then update the `BUFFERSIZE` and `REFRESH` parameters. I suggest to set `REFRESH` to the playback length of a segment, so that after playing one segment the HLS player updates its buffering parameters.

The two parameters together control the downloading behavior of the HLS player by precisely adapting the HLS player buffer size according to the schedule. This indirectly influences *when* an HLS player can download a segment.

Another property of an HLS stream that the scheduling algorithm needs to decide is *which quality* to download. In the case of multi-variant HLS streams, the player tries to download the segments in the quality it prefers by doing its own local measurements. But the schedules also include the HLS video quality for each segment, selected from the available HLS variants.

Every time the HLS player requests an HLS master playlist the anticipatory scheduling controller downloads the playlists of the scheduled variants and creates a merged single-variant playlist out of the multi-variant playlist.

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-TARGETDURATION:10
#EXT-X-BUFFERSIZE: 2          ← new tag for buffer size
#EXT-X-REFRESH:10             ← new tag for refresh interval
#EXTINF:10,
http://hostname/med/001.ts    ← from medium quality variant
#EXTINF:10,
http://hostname/med/002.ts    ← from medium quality variant
#EXTINF:10,
http://hostname/med/003.ts    ← from medium quality variant
#EXTINF:10,
http://hostname/med/004.ts    ← from medium quality variant
#EXTINF:10,
http://hostname/low/005.ts    ← from low quality variant
#EXTINF:10,
http://hostname/med/006.ts    ← from medium quality variant
#EXTINF:10,
http://hostname/high/007.ts   ← from high quality variant
#EXTINF:10,
http://hostname/high/008.ts   ← from high quality variant
#EXT-X-ENDLIST
```

Figure 7.2: Merged single-variant HLS playlist with `REFRESH` and `BUFFERSIZE` extensions; colors indicate the different variants

As shown in Figure 7.2, segments from different variants are selected and placed in a new single variant playlist according to the download schedule. Only the merged single-variant playlist is then returned to the HLS player. The decision which quality to download is hereby made by the anticipatory scheduling controller and not by the HLS player anymore. The merged playlist contains the `REFRESH` and `BUFFERSIZE` parameters. Each time an HLS player refreshes an HLS playlist, it receives a new value for the `BUFFERSIZE` parameter. A list of example values for the `BUFFERSIZE` in Figure 7.2 for each time slot are listed in Table 7.1 for each refresh of the playlist.

Table 7.1: `BUFFERSIZE` values for time slots

Time slot	1	2	3	4	5	6	7	8
`BUFFERSIZE`	2	3	3	0	0	1	1	1

Through both the `BUFFERSIZE` and the `REREFSH` mechanisms, the buffer size (*when* to download) and pre-selection of variants (*which quality* to download) can be controlled. Thus, anticipatory buffering and quality selection based on the previously described algorithms can be performed by simply extending the HLS protocol with two small extensions to the playlist parameters.

7.2.2 Testbed

In order to analyze the algorithms and to test the HLS protocol extension in a real system, I have set up a testbed that allows to run extensive tests with real hardware and compare the results of these tests with the simulations.

The testbed is based on the architecture explained before. The UEs are smartphones and tablets with a customized Android operating system and a modified VLC video player. The modifications enable VLC to parse the additional playlist parameters and adapt its buffer size accordingly. It also outputs extended information about the buffer size and the downloaded segments which is used for our measurements.

The radio access network in the testbed is implemented with 802.11g wireless LAN [WLA12] without any modifications and four access points. Scheduling happens on the application layer, thus changes to the wireless MAC are not necessary. The access points are normal PCs with wireless LAN cards and Linux with hostapd running on them.

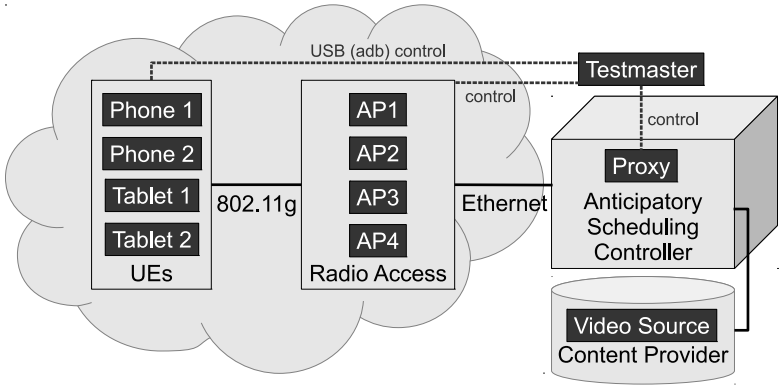
A fifth PC serves as central control and measurement unit and runs the anticipatory scheduling controller. All phones are connected to this PC via USB and are controlled with the Android debug bridge (ADB). The access points are controlled via an SSH connection. With the ADB we execute arbitrary shell commands on the phones and emulate simple user interaction like starting or stopping a video stream. No data is transmitted via USB; it only serves to make experiments repeatable.

The resulting testbed architecture and setup can be seen in Figure 7.3.

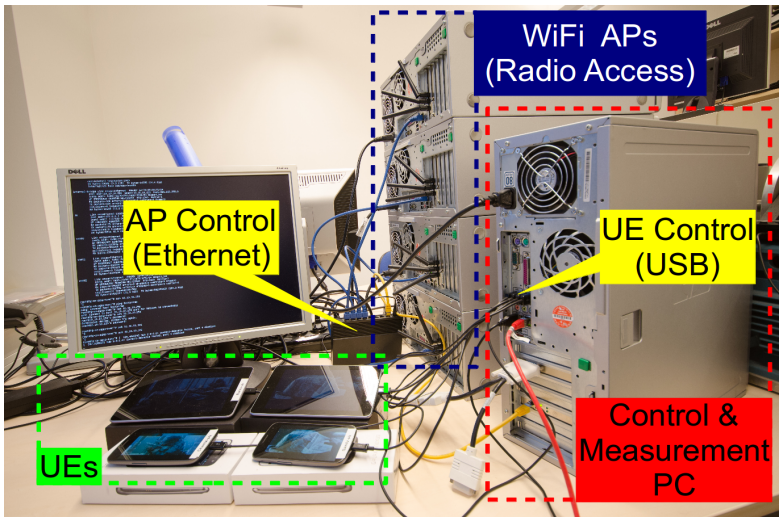
For the HLS video stream content I use the publicly available movie "Tears Of Steel"[1] which I converted to a HLS stream using the VLC framework. The segments and playlists are served by an unmodified Apache webserver.

The anticipatory scheduling controller, which intercepts and modifies the playlist requests from the UEs, is implemented as a transparent HTTP proxy using the Python framework Twisted [Twi]. The access points redirect all traffic coming from the UEs to the proxy thus it is not necessary to change any configuration on the UEs.

[1] `https://mango.blender.org/`



(a) Testbed Architecture



(b) Testbed Setup

Figure 7.3: Testbed

I wanted to be able to run a lot of repeatable and comparable tests, which is why the movement of the UEs is emulated and not done physically. Movement emulation works by limiting the link speed and enforcing handovers between access points. I achieve this by using standard traffic shaping capabilities of Linux on the access points and on the phone. From a predefined scenario I get the data rate for every UE and base station per time slot. These values are then set as speed limits on the access points at the corresponding time. Handover events between the access points are also pre-calculated based on the scenario and then triggered on the phones. With this setup I can run tests without the need to physically move the UEs.

I automatically start the video stream via the ADB connection to the phones and collect information about the streaming (i.e., *when* a segment has been actually downloaded in *which quality*). The results returned by the testbed runs are in the same format as the simulation results and allow a direct comparison.

7.3 Evaluation

In this evaluation, I assume a perfect prediction of future data rates and only compare the algorithms with perfect prediction from Chapter 4 to avoid any side effects from prediction errors.

The plots in Figures 7.4 and 7.5 show a comparison between simulation results with the testbed scenario and the measurements obtained from the testbed. The results from the simulation are plotted with a solid line and the testbed measurements with a dashed line, both using the same markers to distinguish between the schedulers. The plots are separated for the greedy algorithms and the anticipatory solutions for better readability.

Ideally, the simulation results and the testbed measurements should be identical. Differences in the results are due to the following effects, which are present in the testbed but not considered in the simulation:

- *Continuous time*
  The simulation is based on a discrete time model with time slots, whereas the testbed runs in real time. In order to compare the simulation and testbed results, the measurements are converted to discrete time. This, for example, implies that a segment that is actually downloaded after 61 seconds, but should have been downloaded at or before 60 seconds is treated as late as a segment that is downloaded after 69 seconds.

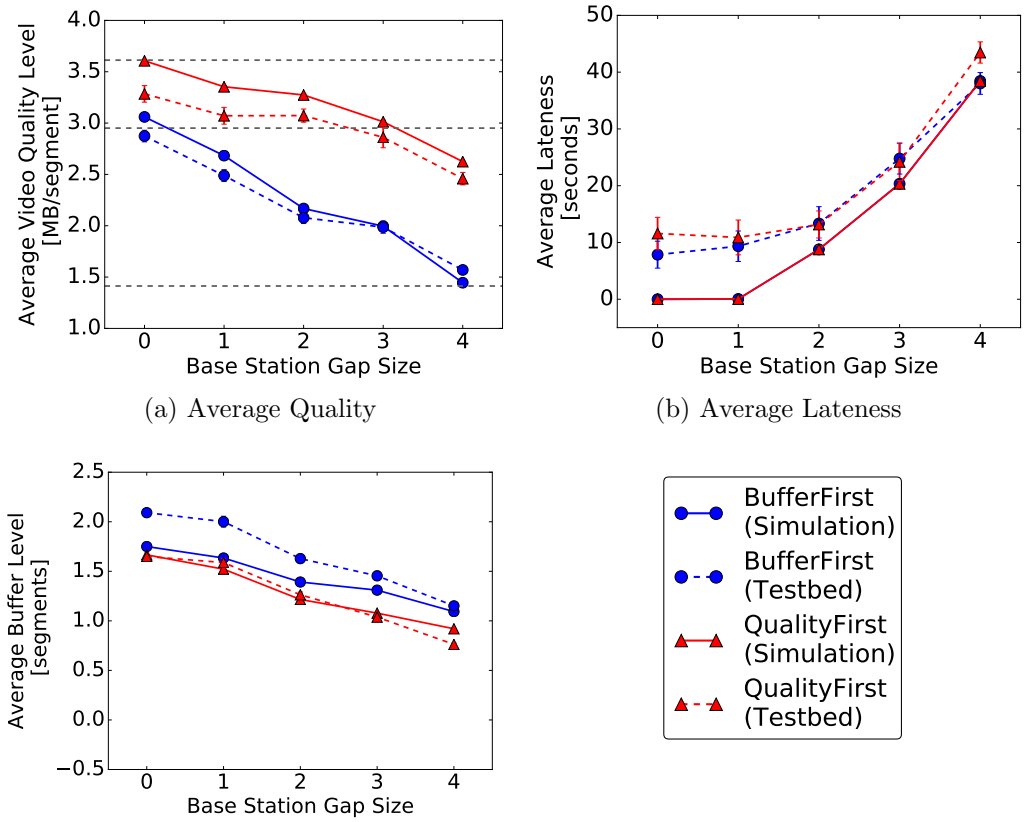
- *Network protocol side effects*
  The simulation does not consider underlying network protocols for transporting the HLS segments. In contrast to that the testbed uses real HLS over TCP/IP over 802.11g wireless LAN with its own wireless resource scheduler. I am only sure that the data rate limits we use in the calculation of the schedules are not *exceeded*, but we cannot ensure that they actually *fully achieved* in the testbed. Both TCP congestion control and the wireless resource scheduler can influence the actual data rates in the testbed, resulting in longer segment downloads, which are then treated as late.

- *Video player issues*
  In case the video player in the testbed has issues while decoding the video, the timing between the downloads from the player and the schedule can be disturbed. For example, if VLC decides to skip frames from the video the playback runs ahead of the calculated schedule and subsequent segments are needed for playback before their download was scheduled to be complete. This can happen because the video player runs on a real Android device and has to share the CPU with the system and background processes.

The measurement results for the average video quality in Figures 7.4a and 7.5a show small differences between the simulation and testbed. These small differences are a result from slightly delayed segment downloads or issues with the video player as previously before. This indicates that the extension to the HLS standard (Section 7.2.1) for quality selection works in the testbed implementation as well as expected based on the simulation.

Figures 7.4b and 7.5b show the average lateness in the testbed. The measurement results for the greedy schedulers show only small differences compared to the simulation but the measurement results for the MIQCP and the FILL scheduler show a significantly higher lateness for the testbed. I discovered that this is due to the buffer minimization in these two schedulers: being forced to use a low buffer level, and ideally not buffering any segments at all if it is not necessary, makes the video player more susceptible to the timing side effects I previously listed.

The results for the average buffer fill level in Figures 7.4c and 7.5c show only a small difference between the simulation results and the testbed measurements. This indicates that the extension to the HLS standard (Section 7.2.1) for the buffer level works in the testbed implementation as well as expected based on the simulation.

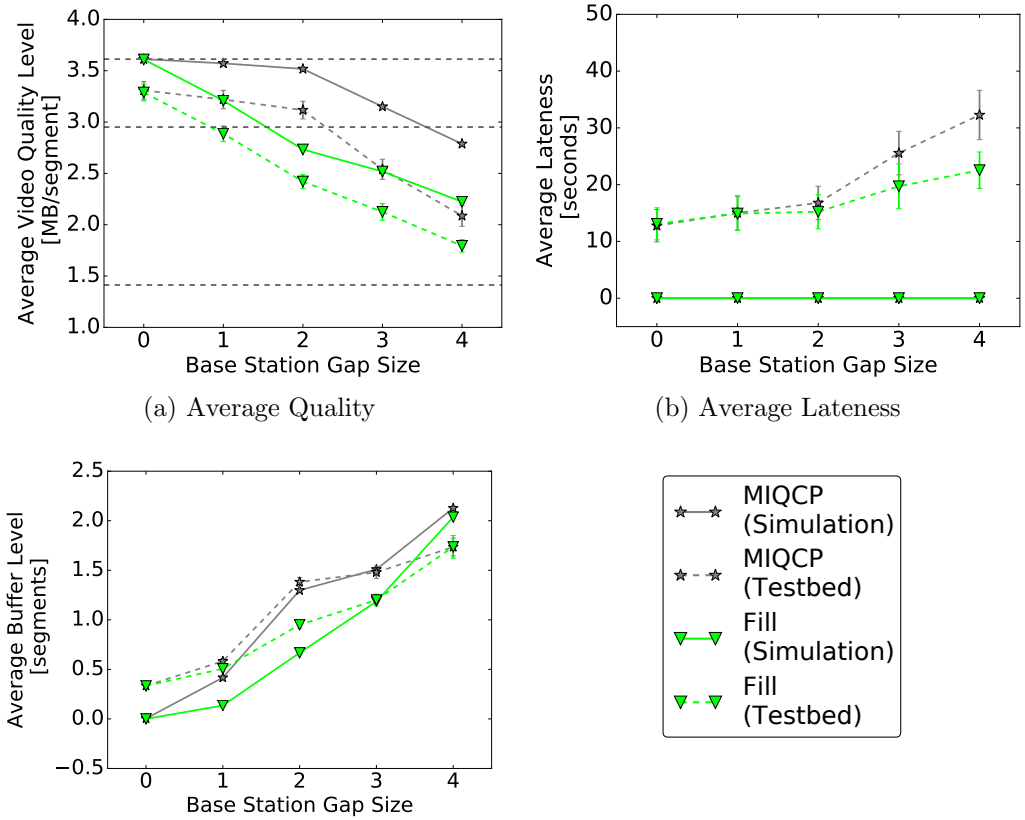


(a) Average Quality

(b) Average Lateness

(c) Average Buffering

Figure 7.4: Testbed measurement results (dashed lines) compared to simulation results (solid lines), greedy algorithms



(a) Average Quality

(b) Average Lateness

(c) Average Buffering

Figure 7.5: Testbed measurement results (dashed lines) compared to simulation results (solid lines), MIQCP and Fill algorithm

7.4 Summary

In this chapter, I have shown a prototype implementation for anticipatory download scheduling. The prototype implementation includes a backwards-compatible extension to the HLS protocol that can also be used in a real-world deployment of anticipatory download scheduling.

Based on my evaluation of the prototype and taking into account the side effects from the testbed setup, I can sum up that the testbed implementation of the anticipatory scheduling works as forecasted by the simulation results. This agreement of results between two different and independent evaluation methodologies lends considerable evidence to the utility and feasibility of the proposed anticipatory scheduling scheme.

8

State of the Art & Related Work on Backhaul Network Reconfiguration

My work on *backhaul network reconfiguration* is based on previous work I did together with Thorsten Biermann, which I introduce in Section 8.1. The approaches presented in this thesis are also strongly related to the CROWD Controller Architecture (CCA) as part of the EU FP7 project *CROWD*. I explain the CCA in Section 8.2. After that I give an overview of related work in Section 8.3.

8.1 Backhaul Network Reconfiguration for CoMP

In his PhD thesis [Bie12], Thorsten Biermann investigated the requirements and constraints on backhaul networks from implementing Coordinated MultiPoint transmission and reception (CoMP). Together with him I developed an approach [DBKK12, DBK13] to check for Coordinated Base Station Sets (CBSs) and whether they are feasible with respect to the backhaul network or not. We implemented this approach both as a linear optimization problem and as a heuristic algorithm. Both are the basis for my approach in Chapters 9 and 10.

We have developed the following system architecture for backhaul network reconfiguration, as shown in Figure 8.1. After detecting that CoMP is required in step ①, e.g., because the requested service quality of a User Equipment (UE) cannot be satisfied, the desired CBSs are determined. This is done in step ② and requires information about the wireless channel conditions, like Received Signal Strength Indication (RSSI) measurements. There are several methods to determine desired CBSs [PGH08]. These CBSs are used together with the current status of the backhaul network to calculate which part of the desired CBSs is feasible. This happens in step ③ and is done by a heuristic algorithm. Now if the backhaul capabilities are sufficient to establish the complete desired CBS, CoMP

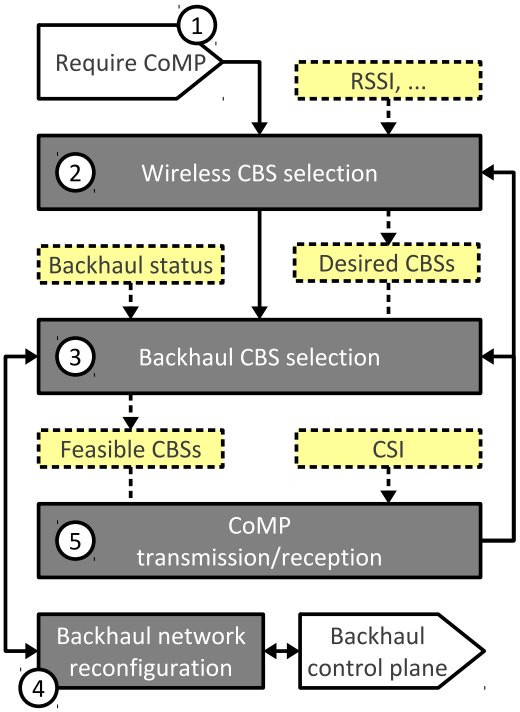


Figure 8.1: Overall CBS selection/reconfiguration system architecture

is conducted immediately in step ⑤. Otherwise, backhaul network reconfiguration is triggered to improve the number of feasible CBS.

Block ④ needs to decide which actions to take in the backhaul network to improve the number of CBS. This decision is made based on two kinds of information. First, the output of the CBS selection heuristic in step ③, which contains the reasons why certain Base Stations (BSs) cannot be in a feasible CBS. And second, information about the backhaul network itself, e.g., which parameters can be changed for which links. This information is provided by the network control plane. Now, by matching these two things, the backhaul network can be reconfigured via its control plane to eliminate the bottlenecks that prevent desired CBSs. This mechanism can also be used to reduce the backhaul network resources again when they are not needed, e.g., to save energy.

After a reconfiguration step, the CBS selection in step ③ has to be done again to check the improved CBS feasibility. If necessary, this loop can be executed multiple times until the desired CBS becomes feasible or a termination condition, like a maximum number of performed reconfigurations, is reached.

This overall procedure has to be repeated whenever the desired CBS, i.e., the long-term wireless channel conditions, or the CBS feasibility, i.e., the backhaul network load and latency, change significantly. In these cases, the corresponding CBS selection steps have to be done again, as indicated by the arrows on the right side of Figure 8.1.

This overall architecture clearly separates *selection* of CBS and *reconfiguration* of the backhaul network in two individual steps (③ and ④). This split reduces the

number of feasible CBSs compared to an integrated approach which I present in Chapter 9. The evaluations from Thorsten Biermann and myself [DBK13] also show that this split approach cannot be applied successfully in scenarios with a high density of UEs. My integrated approach removes this limitation as I show in Chapter 10.

8.2 CROWD Controller Architecture

I developed my approach for *backhaul network reconfiguration* as a part of the CROWD Controller Architecture (CCA) [AACdlO+13a, CMD+13]. The CCA is a mobile access network architecture for dense wireless access networks. It is developed based on the Software-Defined Network (SDN) paradigm (see Section 2.1.3) and supports software-defined base station coordination (see Section 2.3.2). In this section I present a high-level overview of the CCA.

The CCA encompasses both Long Term Evolution (LTE) and WiFi cells, which are the technologies expected to have the highest penetration in future mobile access networks. It is based on the assumption that all network elements belong to the same administrative domain (e.g., network operator). Thus, security measures for preventing malicious access of the control functions and avoiding unauthorized disclosure of sensitive information from customers are neglected so far.

The architecture is structured into two logical tiers: *districts* with a limited but fine-grain scope for short time scales and *regions* with a broader but more coarse-grain scope for long time scales.

A *district* consists of BSs, i.e., LTE eNodeBs and WiFi Access Points (APs), as well as interconnecting backhaul links that are assumed, without loss of generality, to be reconfigurable via an open protocol, e.g., OpenFlow. Operation within a district is optimized by applications connected to a so-called CROWD Local Controller (CLC) via a set of Application Programming Interfaces (APIs), referred to as northbound interfaces in the SDN terminology. There are two types of northbound APIs:

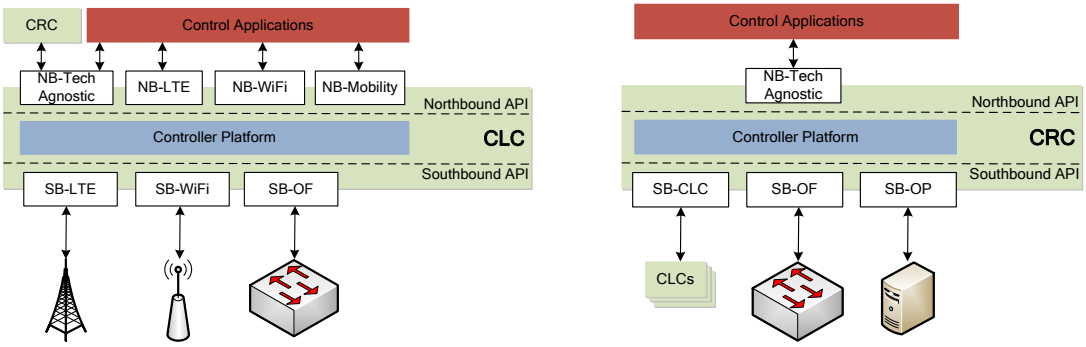
1. *Technology-specific APIs*, which expose fine-grained details as acquired from the BSs (e.g., sub-frame utilization in LTE) and offer methods which are only valid for the specific communication protocol.

2. *Technology-agnostic APIs*, which expose abstract and aggregated data (e.g., average node utilization) and offer generic modifiers which may be valid for a wide range of technologies and capabilities (e.g., switch off a node).

Any application can connect to one or more APIs, depending on its optimization goals and requirements. Based on the technologies present in the district, the CLC can access different southbound interfaces for LTE and WiFi to control the wireless operations and OpenFlow for controlling the backhaul network. An overview of the CLC interfaces is shown in Figure 8.2a.

A special use of the technology-agnostic API is to connect a CLC to its higher-level controller, the so-called CROWD Regional Controller (CRC), which operates

inside a *region*. The region is defined as a logical area including several districts in which technology-agnostic applications are executed for longer scale optimizations, compared to the CLC applications. Regional optimization is proposed to compensate for sub-optimal choices which may be taken at district level because of the myopic sight of the local controllers. An overview of the CRC interfaces is shown in Figure 8.2b. The CRC only exposes a technology-agnostic interface on its northbound API for regional control applications. The southbound API of the CRC includes a specific interface to control CLCs inside the region and interfaces to the OpenFlow backhaul network and for information exchange with the network operator infrastructure.



(a) CLC (b) CRC

Figure 8.2: CCA controllers: architecture and interfaces [AACdlO+13a]

A simplified example for the interaction between the CLC and the CRC is the following: From the point of view of an application running within the CLC with the goal of minimizing energy consumption, an "optimal" choice could be forcing all user terminals to associate to BSs outside of the district and switching off all the BSs inside the district. Such a drastic decision would be obviously sub-optimal from a broader network viewpoint, thus any CRC application aiming to minimize energy consumption would certainly override it.

A diagram of the overall network architecture is shown in Figure 8.3, which also shows the southbound interfaces between the CLC and the BSs and backhaul, as well as some key interconnections with new and existing network elements.

For example, in the case of LTE, the eNodeBs have a split connection: the control path, i.e., via the 3GPP S1-MME and X2 interfaces, goes entirely through the CLC, whereas the data path is directed to the Distributed Mobility Management (DMM) gateway, which is a novel element proposed as a part of the CCA. Therefore, a CLC application can intercept the communication between the eNodeB and the Mobility Management Entity (MME) and anticipate/override intra-district mobility decisions. Note also that the CCA proposes the use of the X2 interface for collecting fast and detailed measurements from the LTE eNodeBs, since it already supports a wide range of data, even though the standard assumes that information is exchanged between peer eNodeBs.

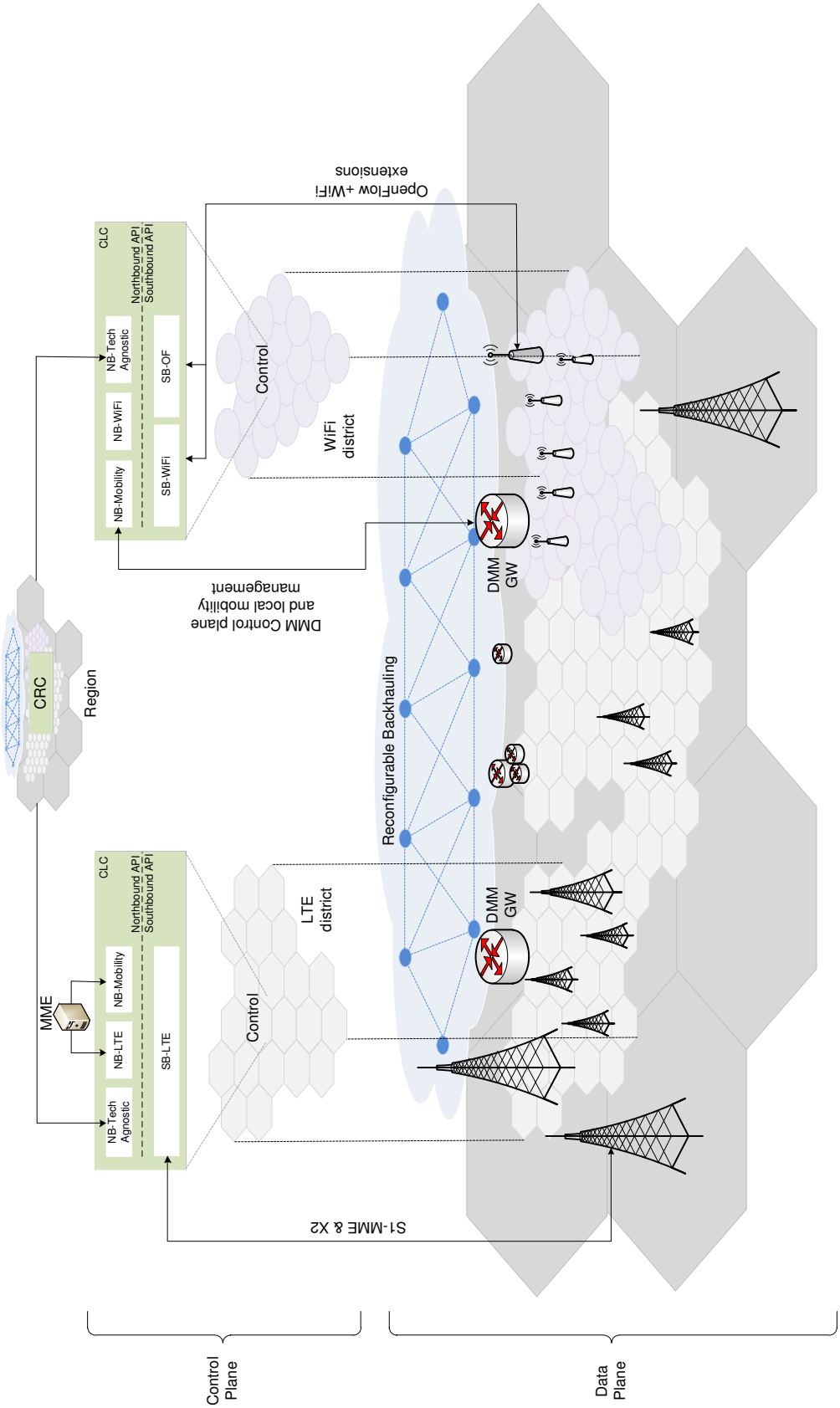


Figure 8.3: CROWD network architecture [AACdlO+13a]

8.3 Related Work

Existing evaluations show how a limited backhaul network reduces the efficiency of CoMP [BSC+11, SSPS09] but the proposed approaches only try to improve the CBS selection on the wireless side to incorporate the backhaul limitations.

The problem of a limited backhaul network together with implementing CoMP has been addressed in existing work:

Soliman et al. [SNK13] analyze how the backhaul resources have to be shared to achieve a feasible data exchange between coordinated BSs but their model only considers two BSs. Their approach is designed to *react* to a constrained backhaul, not to *reconfigure* it.

De Domenico et al. [DSK13] and Olmos et al. [OFGZ13] investigate the assignment of users to coordinated base stations but considering also the backhaul network capacity. With this approach they also *react* to a constrained backhaul and ensure that the capacity of the backhaul is not exceeded by the user assignment, but they do not *reconfigure* the backhaul network as it is only considered as a simple capacity constraint.

The approach presented by Zhao et al. [ZL12] aims at limiting the coordination of transmissions to minimize the required backhaul load and does not take a constrained backhaul into account at all.

Marsch et al. [MF07] developed a framework to optimize the performance of wireless coordination under a constrained backhaul by selecting only a subset of users for coordinated transmissions. This is very similar to the selection of CBSs, but does not consider any *reconfiguration* of the backhaul network. Also their backhaul model is very generic as they only investigate a simple full mesh backhaul topology without any latency constraints.

Zhang et al. [ZYM12] investigate effects of a constrained backhaul on different BS coordination schemes and propose a selection method to choose the best coordination scheme for the available backhaul capacity. Their approach but do not include the selection of feasible CBSs.

Bartelt et al. [BFW+13] analyze the challenges for backhaul networks arising from a cloud-based RAN and conclude that optical networks are a promising technology for Cloud RAN backhauls, if the data rate and latency demands are included into a joint optimization of radio access and backhaul. My combination of CBSs selection and backhaul network reconfiguration is exactly such a joint optimization.

Liu et al. [LSJ+13] investigate a dynamic backhaul network for the dynamic assignment of base stations to controllers with a reconfigurable backhaul architecture, including a small testbed evaluation. For their approach they consider the backhaul network as an virtual overlay network on top of a static physical backhaul network. This makes the reconfiguration of the backhaul network simpler compared to my approach for Wavelength-Division-Multiplexed Passive Optical Network (WDM-PON) backhauls.

9

Backhaul Network Reconfiguration

In this chapter, I introduce my approach for *backhaul network reconfiguration*. As introduced before in Section 1.3.2, wireless coordination is only feasible if the backhaul network provides sufficient capacity and low latency between the Base Stations (BSs) and the controller, which is responsible for managing the coordination scheme. Thus the goal of the *backhaul network reconfiguration* is to find a suitable allocation of backhaul resources, in particular wavelengths in Wavelength-Division-Multiplexed Passive Optical Networks (WDM-PONs) backhaul networks, for each desired Coordinated Base Station Set (CBS) to make it feasible.

I provide a formal problem description in Section 9.1 and describe an optimal solution in Section 9.2 and a heuristic algorithm in Section 9.3. In Section 9.4 I present evaluation results for both solutions, including results about the energy efficiency of the approach.

9.1 Problem Description

For each desired CBS two decisions have to be taken:

1. Which node should be the controller for the CBS?
2. Which links and wavelengths in the backhaul should be used to form routing paths to exchange data between the controller and the BSs in the CBS?

A CBS can only be considered feasible if at least one feasible controller node can be found. Finding a controller node is only feasible if at least one feasible wavelength assignment for all routing paths to the BSs exists in the backhaul network. A wavelength assignment is feasible if either there is sufficient free capacity on already assigned wavelengths or new wavelengths can be assigned to provide sufficient capacity.

I consider the backhaul network as a directed graph G with vertices V and edges E. In my model all vertices are BSs, which means they can all be part of a CBS and are all able to act as a controller. Without loss of generality, my model can easily be extended with nodes that cannot be part of a CBS or cannot act as a controller (e.g., switches or routers). All desired CBSs are defined as sets W_i of vertices from V. For each CBS i there is a required capacity $b_{\text{req}}(v, i)$ for each vertex v and a maximum latency $l_{\max}(i)$ between the controller and all BSs in the CBS. For all edges there is a set of unassigned wavelengths K that can be assigned to each edge. Consistent with the requirements from the CBSs, each edge (u, v) has a latency $l_{\text{cap}}(u, v)$ and a capacity per wavelength $b_{\text{cap}}(u, v, k)$ for each wavelength k.

In order to consider a CBS feasible regarding the backhaul network, one vertex c_i has to be selected as the controller for CBS i. Additionally, routing paths from the controller c_i to all vertices of the CBS have to be computed. Each routing path is a list of consecutive edges and an assigned wavelength per edge, such that (a) the capacity on all assigned wavelengths is greater than or equal to the required capacity of the BS and (b) the total latency of the path does not exceed the maximum latency for the CBS. If a controller and routing paths are found, the CBS is considered as feasible, otherwise as infeasible.

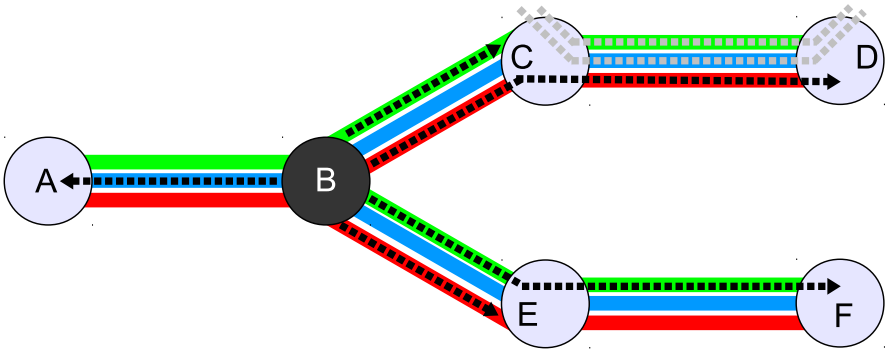


Figure 9.1: Example: controller selection and wavelength assignment

Figure 9.1 shows a simple example: vertex B is selected as the controller and routing paths from vertex B to vertices A, C, D, E and F are assigned. Assuming the capacity requirement for each vertex is equal to the capacity of a whole wavelength, the controller at vertex B is the only feasible choice. The wavelength assignment for the paths from B to A, C, E and F is arbitrary, but the assignment shown in the figure for the link from C to D is the only feasible assignment because the other two wavelengths are already used for routing paths of another CBS.

9.2 Optimal Solution

The Integer Linear Program (ILP) described in this section provides a precise specification of the problem outlined in the previous section and is amenable to an automatic solution.

9.2.1 Integer Linear Program

The optimization problem is an ILP and takes the parameters listed in Table 9.1 as inputs. It correspond to the problem definition in Section 9.1.

Table 9.1: ILP input parameters

V	set of Base Stations (BSs)
E	set of links with $E \subseteq V \times V$
K	set of wavelengths
W_1 ... W_n	n input CBSs with $W_i \subseteq V$
$b_{\text{req}}(v, i)$	required capacity for BS v in CBS i
$b_{\text{cap}}(u, v, k)$	capacity for wavelength k on link from BS u to BS v
$l_{\max}(i)$	maximum round-trip latency between controller and BS in CBS i
$l_{\text{cap}}(u, v)$	latency on a link from BS u to BS v

The variables listed in Table 9.2 are used to represent the decisions about the controller placement and the backhaul configuration for each CBS.

The following constraints have to be ensured. Some constraints use a big-M constant denoted as $\mathcal{M}$, which is a very large constant.

All BSs of one CBS need to be the end of a path (Eq. 9.1) and all paths need to start at a controller (Eq. 9.2).

$$\sum_{\substack{s \in V, s \neq d, \\ (u,d) \in E, o \in K}} f_{s,d,u,d,o,i} = w_i, \forall d \in W_i, i \in \{0, ..., n\} \tag{9.1}$$

$$\sum_{\substack{s \in V, s \neq d, \\ (s,v) \in E, o \in K}} f_{s,d,s,v,o,i} = w_i, \forall d \in W_i, i \in \{0, ..., n\} \tag{9.2}$$

Table 9.2: ILP variables

$w_i \in \{0,1\}$	determines whether CBS i is fully established
$c_{s,i} \in \{0,1\}$	determines whether BS s is a controller for CBS i
$k_{u,v,o} \in \{0,1\}$	determines whether wavelength o is used on a link from BS u to BS v
$j_o \in \{0,1\}$	determines whether wavelength o is used anywhere
$p_{s,d,o,i} \in \{0,1\}$	determines whether wavelength o is used on the path between s and d in CBS i
$q_{d,i} \in \mathbb{N}$	determines the number of wavelengths used for BS d in CBS i
$f_{s,d,u,v,o,i} \in \{0,1\}$	determines whether link u to v using wavelength o is included in the path from a controller at BS s to a BS d for CBS i

For all other nodes (non-controllers, BSs not in the CBS) the ingress and egress paths have to be balanced (Eq. 9.3).

$$\sum_{(u,v)\in E, o\in K} f_{s,d,u,v,o,i} = \sum_{(v,w)\in E, o\in K} f_{s,d,v,w,o,i}, \tag{9.3}$$
$$\forall s \in V, d \in W_i, v \in V, i \in \{0,...,n\}, s \neq v, d \neq v, s \neq d$$

Whenever a wavelength is used for a path, the decision variable for this wavelength has to be activated (Eq. 9.4).

$$\mathcal{M} \cdot k_{u,v,o} \geq \sum_{s\in V, d\in W_i, s\neq d} f_{s,d,u,v,o,i}, \forall (u,v) \in E, o \in K, i \in \{0,...,n\} \tag{9.4}$$

Also the decision variable for the controller placement has to be activated for the selected node for the controller (Eqs. 9.5 and 9.6).

$$\mathcal{M} \cdot c_{s,i} \geq \sum_{\substack{d\in W_i, s\neq d,\\ (u,v)\in E, o\in K}} f_{s,d,u,v,o,i}, \forall s \in V, i \in \{0,...,n\} \tag{9.5}$$

$$c_{s,i} \leq \sum_{\substack{d\in W_i, s\neq d,\\ (u,v)\in E, o\in K}} f_{s,d,s,u,o,i}, \forall s \in V, i \in \{0,...,n\} \tag{9.6}$$

Furthermore, no path must end at a controller (Eq. 9.7).

$$\sum_{\substack{(u,s)\in E,\\ o\in K}} f_{s,d,u,s,o,i} = 0, \forall s \in V, d \in W_i, s \neq d, i \in \{0,...,n\} \tag{9.7}$$

For each CBS determined to be feasible the decision variable has to be activated (Eq. 9.8).

$$w_i = \sum_{s\in V} c_{s,i}, \forall s \in V, i \in \{0,...,n\} \tag{9.8}$$

For the variables j, p and q the number of active wavelengths has to be counted (Eq. 9.9, 9.10 and 9.11).

$$\mathcal{M} \cdot j_o \geq \sum_{\substack{u \in V, v \in V, \\ (u,v) \in E}} k_{u,v,o}, \forall o \in K \tag{9.9}$$

$$\mathcal{M} \cdot p_{s,d,o,i} \geq \sum_{\substack{u \in V, v \in V, \\ (u,v) \in E}} f_{s,d,u,v,o,i}, \tag{9.10}$$

$$\forall s \in V, d \in W_i, i \in \{0, ..., n\}, o \in K, s \neq d$$

$$q_{d,i} = \sum_{\substack{s \in V, o \in K, \\ s \neq d}} p_{s,d,o,i}, \forall d \in W_i, i \in \{0, ..., n\} \tag{9.11}$$

Each wavelength must only be used for one direction of a link between two nodes (Eq. 9.12).

$$k_{u,v,o} + k_{v,u,o} \leq 1, \forall (u,v) \in E, (v,u) \in E, o \in K \tag{9.12}$$

For each wavelength on a link the maximum capacity limit has to be ensured (Eq. 9.14).

$$\sum_{\substack{i \in \{0,...,n\}, s \in V, \\ d \in W_i, s \neq d}} f_{s,d,u,v,o,i} \cdot b_{\text{req}}(d,i) \leq b_{\text{cap}}(u,v), \tag{9.13}$$

$$\forall (u,v) \in E, o \in K$$

For each path between the controller and each BS in the CBS the limit on the maximum latency has to be ensured (Eq. 9.15).

$$\sum_{\substack{(u,v) \in E, \\ o \in K}} f_{s,d,u,v,o,i} \cdot (l_{\text{cap}}(u,v) + l_{\text{cap}}(v,u)) \leq l_{\max}(i), \tag{9.14}$$

$$\forall i \in \{0, ..., n\}, s \in V, d \in W_i, , s \neq d$$

The objective function (Eq. 9.16) uses the following weight factors: m_b for established CBSs, m_c for used wavelengths per BS in CBS, m_g for globally used wavelengths and m_k for used edges.

$$\begin{aligned} \text{maximize: } & m_b \cdot \sum_{i \in \{0,...,n\}} w_i \\ & - m_c \cdot \sum_{d \in W_i, i \in \{0,...,n\}} q_{d,i} \\ & - m_g \cdot \sum_{o \in K} j_p \\ & - m_k \cdot \sum_{(u,v) \in E, o \in K} k_{u,v,o} \end{aligned} \tag{9.15}$$

I use these weight factors to define the following lexicographical ordering:

1. Maximize established CBSs
2. Minimize the number of used wavelengths per CBS
3. Minimize the overall number of used wavelengths
4. Minimize used links

To establish a correct lexicographical order, the exact values of the weight factors have to be adapted to the number of CBSs, wavelengths and links.

9.2.2 Complexity

Biermann et al. have shown that CBS selection without *desired* CBSs as input is an NP-hard problem [BSWK11]. The main difference to this former problem is the additional input of *desired CBSs* and the changed objective: instead of searching for an arbitrary set of CBSs that covers all BSs in the network, I look for a CBS that includes all or as many as possible *desired* CBSs, while adhering to the backhaul network constraints for capacity and latency. There are no further restrictions how *desired* CBSs are structured, each *desired* CBS is just a set of BSs. Additionally I consider the assignment of wavelengths in the backhaul network, which was also not considered by Biermann et al.

Without considering the wavelength assignment and only considering the determination of feasible routing paths from the controller to all BSs in a *single* CBS, the problem can be interpreted as an extended variant of the bandwidth-delay-constrained least-cost multicast routing problem for each CBS with the controller as source and all BSs in the CBS as destinations. This problem is known to be NP-hard without a known Polynomial-Time Approximation Scheme (PTAS) and it can only be solved heuristically [FMHG08, SRV02, ZBE01].

In the problem I presented in this chapter I additionally consider *multiple* CBSs and the assignment of wavelength and thus conclude that no PTAS for my problem exists.

9.3 BFS Algorithm

Because the ILP cannot be solved for instances with a realistic size for real-world deployments, a heuristic approach is required. I developed an algorithm that solves the problem with a modified Breadth-First-Search (BFS), based on previous work together with Thorsten Biermann [DBK13, DBKK12]. The algorithm in previous work separated the selection of CBSs and the assignment of backhaul resources in two different steps and did not consider the assignment of generic capacities instead of individual wavelengths.

The core concept of the algorithm is to start a BFS individually for each CBS from every BS in the backhaul network, which yields a tree for each start BS. These trees are then checked in an iterative process whether they contain the

desired CBS and whether they obey the constraints on capacity and delay. If the CBS is matched and no constraints are violated, the root BS of the tree is a potential controller for the CBS. After that the routing paths between the controller BS and all BSs from the CBS are mapped to wavelengths.

In contrast to the optimization problem where all CBSs are handled simultaneously, the heuristic algorithm handles the CBSs one by one. If the CBSs had an assigned priority, this priority could be used to order the CBSs for processing. As I do not have any priorities in my model, I use a random order.

9.3.1 Inputs

The inputs to the algorithm are the backhaul network as an annotated graph $G = (V, E)$, where each vertex corresponds to a node (e.g., BS) in the backhaul network and each edge to a link between two nodes with a set of wavelengths K and the desired CBS W_i with $W_i \subset V$. The annotations include the capacity per wavelength $b_{\text{cap}}(u, v, k)$, the edge latency $l_{\text{cap}}(u, v)$, the required capacity per vertex in CBS i $b_{\text{req}}(v, i)$, and the maximum round-trip latency $l_{\max}$ between two cooperating BSs.

9.3.2 Algorithm Implementation

An overview of the heuristic is depicted in Figure 9.2 and the individual steps are explained below.

1. Maximum-Path BFS The heuristic performs a modified BFS, separately from each vertex of the graph as the root. Whenever a new tree edge (u, v) is discovered, the predecessor annotation $p(v)$ for vertex v is set to u and the constraints for the new edge are checked in the following way:

- Is the latency to the new vertex v via the whole path from the root via u and the new edge (u, v) smaller than the maximum round-trip latency?
- Are there any wavelengths on (u, v) with enough free capacity to connect v to the root vertex, if v is part of the CBS?

If either of these checks fails, the new vertex v is not added to the tree. The result of this step is a set T of BFS trees, which only contain vertices that meet the constraints within the tree. This does not yet take into account reciprocal effects between multiple BSs in the CBS.

The implementation for checking the constraints is described in Algorithm 9.1. The input parameters are the new vertex v, its predecessor u, the BFS tree root node s, and the CBS index i.

2. Match CBS The heuristic checks for every BFS tree T_i from step 1. whether it completely contains the desired CBS. The result is a set of BFS trees T that match the desired CBS.

1. **Maximum-Path BFS**
   ▷ start modified BFS from each vertex
   ▷ *output*: BFS tree for each vertex
2. **Match CBS**
   ▷ match BFS trees against CBS
   ▷ *output*: possible candidate BFS trees for CBS
3. **Back-Track BFS Trees**
   a) **Check constraints**
      ▷ recheck constraints on candidate BFS trees
   b) **Wavelength Assignment**
      ▷ determine wavelengths for all candidate BFS trees

   ▷ *output*: possible candidate BFS trees and their wavelength assignment for CBS
4. **Match CBS**
   ▷ match BFS trees against CBS again
   ▷ *output*: confirmed candidate BFS trees for CBS
5. **Find Best BFS Tree**
   ▷ compare candidate BFS trees
   ▷ *output*: best BFS tree for CBS

Figure 9.2: BFS Algorithm

3. Back-Track BFS Trees The constraint checking in step 1. discards vertices based on the latency for the whole paths to the start node and only the link capacity for single links and not for whole paths between the start node and each BS. Thus, the capacity constraints need to be checked again, now taking into account also the capacities on whole paths.

If the constraints are not violated a wavelength assignment for all routing paths between the BSs from the CBS and the BFS tree root has to be determined. Because the required data structures for this step can easily be constructed in the back-tracking phase, it makes sense to combine the back-tracking and the wavelength assignment into one step.

3.a) Check Constraints The goal of this step is to perform a check of the capacity constraints on the whole paths from each BS of the CBS to the BFS tree root, similar to the check in Algorithm 9.1 for the latency constraints. During this step the algorithm does not only determine *if* there is sufficient capacity on a link, but also *which wavelengths* have sufficient capacity. This information is stored in the matrix WL and is needed in step 3.b).

Algorithm 9.1 CHECKPATHCONSTRAINTS(u, v, s, i)

```
p(v) = u
l_sum ← 0
c_max ← 0
for all k ∈ K do // check capacity
  if b_cap(u, v, k) > c_max then
    c_max ← b_cap(u, v, k)
  end if
end for
if c_max < b_req(v, i) then
  return false // not enough capacity
end if
while v ≠ s do // check latency
  l_sum ← l_sum + l_cap(u, v)
  v ← u
  u ← p(u)
end while
if l_sum · 2 > l_max then
  return false // latency too high
end if
return true // all checks passed
```

The implementation of this step is described in Algorithm 9.2. The input parameters are the BFS tree T and the CBS C_i. Whenever a vertex from the CBS violates the capacity constraints on its routing path, the vertex is removed from the BFS tree. If the constraints are fulfilled for all edges on the path, the capacity annotations for all edges on the path are updated. The implementation of UPDATEANNOTATIONS(T) is simple and omitted here. The implementation of the wavelength assignment in ASSIGNWAVELENGTHS$(\mathrm{WL}_{(u,v)}, c)$ is explained in step 3.b) and Algorithm 9.3.

3.b) Wavelength Assignment The input to the wavelength assignment is a BS c from the CBS and a map M that contains, for each edge (u, v), a set of wavelengths $M_{(u,v)}$ with all candidate wavelengths that have enough capacity to fulfill the capacity constraints of c.

To assign one wavelength out of the corresponding set to each edge (u, v), the algorithm iteratively selects the wavelength that can be used on the maximum number of edges and assigns it (MOSTFREQENTLYAVAILABLEWL(M)). This step is repeated for all remaining edges without an assignment until all edges have a wavelength assignment. Because of the check in step 3.a) it is guaranteed that at least one candidate wavelength per edge exists.

4. Match CBS After removing vertices in the previous step, the information on trees matching the CBS is not valid anymore. Thus, step 2. of the heuristic

Algorithm 9.2 BACKTRACKBFSTREES(T, C_i)

```
for all c ∈ C_i do
  s ← true
  M ← MAP()
  v ← c
  u ← p(c)
  while v ≠ s do // check capacity
    M_(u,v) ← SET()
    for all k ∈ K do // check wavelengths
      if b_cap(u, v, k) > b_req(v, i) then
        INSERT(WL_(u,v), k))
      end if
    end for
    if LENGTH(WL_(u,v)) = 0 then
      s ← false // no wavelength with enough capacity available, tree is not
      a valid candidate
    end if
    v ← u
    u ← p(u)
  end while
  if s = true then
    UPDATEANNOTATIONS(T)
    ASSIGNWAVELENGTHS(M)
  else
    REMOVE(T, v)
  end if
end for
```

has to be repeated for the BFS trees reduced in the previous step. If a tree still matches the CBS, the root is a valid candidate for controlling that CBS.

5. Find Best BFS Tree In the final step, the best BFS tree from the remaining candidates has to be determined. The algorithm calculates three different costs per candidate tree and adds them up in a weighted sum, similar to the objective function of the optimization problem in Section 9.2, as

$$n = \sum (w_g \cdot n_g + w_a \cdot n_a + w_l \cdot n_l) \tag{9.16}$$

with

- n_g as the total number of wavelengths used in the tree, with weight w_g
- n_a as the number of wavelengths that have to be assigned additionally for using that tree, with weight w_a
- n_l as the number of used links, with weight w_l

Algorithm 9.3 assignWavelengths(M)

```
while M ≠ ∅ do
  m ← mostFreqentlyAvailableWL(M)
  for all (u, v) ∈ M do
    if m ∈ M_(uv) then
      assign((u, v), m)
      remove(M, (u, v))
    end if
  end for
end while
```

These costs are then weighted and summed up to calculate the total cost n per tree.

The algorithm then selects the tree with the lowest total cost, sets the root BS as the controller, stores the routing paths, and updates the annotations on G for running the next iteration for the next CBS. The precise weight factors depend on the number of CBSs, wavelengths and links.

9.4 Evaluation

The evaluation is twofold: First, I compare the results of solving the optimization problem to using the heuristic algorithm on small instances. Second, I present results for using the heuristic algorithm in larger, real-world size instances.

All evaluations are executed on Intel Xeon X5650 CPUs running at 2.67 GHz. The optimization problem is solved with Gurobi running in single-thread mode for a comparison with the heuristic algorithm, which is implemented in Python. All plots contain confidence intervals at a 95% level, unless they are too small and covered by the plot markers.

9.4.1 Scenario

In all evaluations, a fixed number of BSs are placed on a regular grid, with a mean inter-BS distance along the grid of $\bar{s} = 1000\,\text{m}$ (urban scenario [BAWB13]), and are then shifted in both x and y direction according to a normally distributed random variable with zero mean and standard deviation $\frac{\bar{s}}{8}$.

The backhaul network topology can then be generated either as a *mesh* or as a *tree* topology. In the *mesh* topology, two BSs are connected by a link if their distance is less or equal to $1.5 \cdot \bar{s}$. This value produces a partially connected mesh network; smaller or larger values result in too sparse or too dense topologies, which are not realistic.

For a *tree* topology the BSs in the same area are connected to a common central node according to a WDM-PON splitting factor. All central nodes are located in a central site and are interconnected.

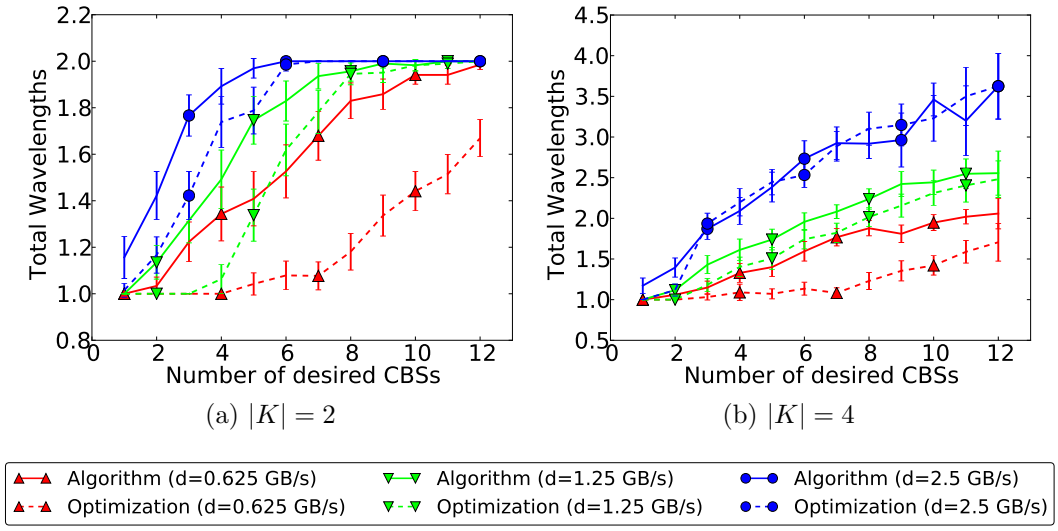
In both cases all links in the backhaul network are assigned the same set of

available wavelengths K and each wavelength is assigned the same fixed capacity of 2.5 Gb/s. The latency for each link is determined by the distance multiplied by 1.45 divided by the speed of light as we are modeling an optical backhaul network.

The desired CBSs are generated by placing circles uniformly at random on the plane covered by the placed BSs. The BSs which are considered as the CBS are all BSs located inside this circle with a given radius. We determine this radius by multiplying the mean inter-BS distance with a factor $r = 1.5$, which results in 5 BSs per CBS on average. The capacity demand d for each BS in the CBS is set to the same value and is either 0.625 Gb/s, 1.25 Gb/s, or 2.5 Gb/s. This implies that at a demand of 2.5 Gb/s one complete wavelength is required to connect a BS to the controller.

9.4.2 Comparison: Optimization vs. Heuristic Algorithm

To compare the results from solving the optimization problem with those of the heuristic algorithm, I choose a scenario with 16 BSs and a mesh backhaul network topology. I use scenarios with 2 or 4 available wavelengths. These scenarios are big enough to see the important effects and small enough to run in reasonable time.



(a) $|K| = 2$ (b) $|K| = 4$

Figure 9.3: Total Wavelengths

For the total number of used wavelengths (Figures 9.3a and 9.3b) the results for $d = 1.25$ and $d = 2.5$ are very similar for the optimization and the heuristic algorithm. Only for $d = 0.625$ the heuristic algorithm performs significantly worse than the optimization, especially for the scenario with 4 wavelengths.

The results for the average number of wavelengths per link (Figures 9.4a and 9.4b) are statistically identical, except for $d = 2.5$. In that case one BS demands a whole wavelength for its connection to the controller and the optimization problem

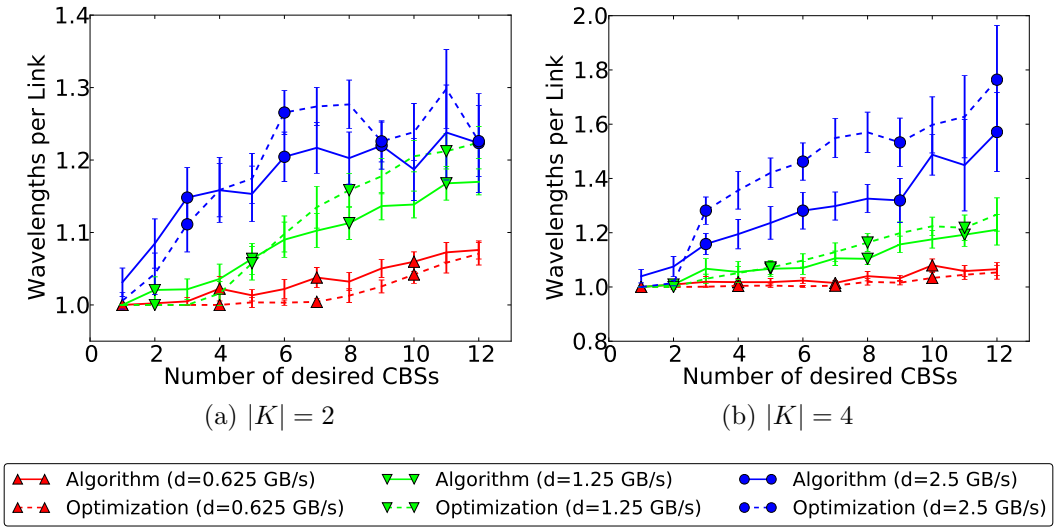


Figure 9.4: Wavelengths per Link

prefers to assign a new wavelength on an already used link instead of routing via a different link as the heuristic algorithm does.

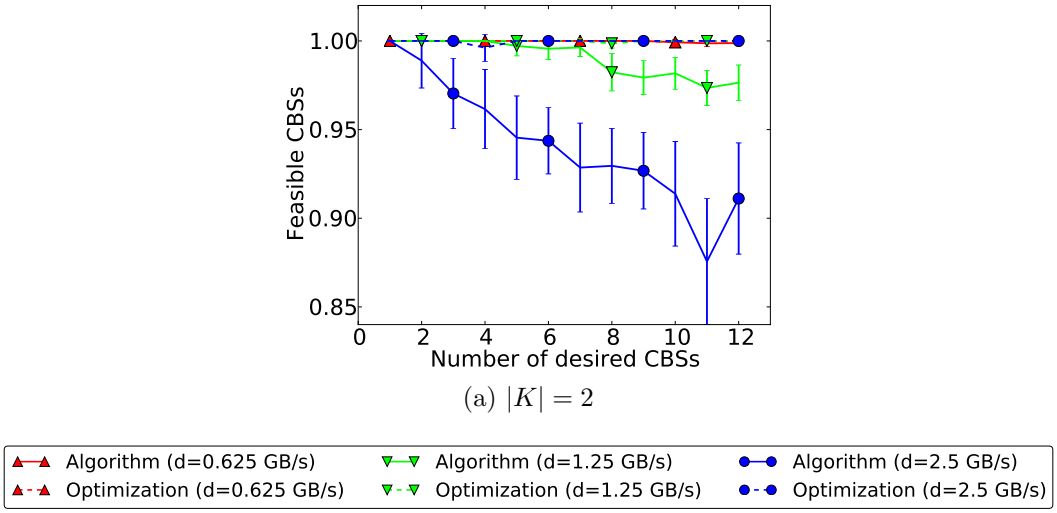


Figure 9.5: Feasible CBSs

Also due to this behavior, the heuristic algorithm is not able to establish all desired CBSs for $d = 2.5$ and $|K| = 2$ as shown in Figure 9.5a. For $|K| = 4$ all CBSs are feasible, thus I omit this plot.

The total execution time for both the optimization and the heuristic algorithm is shown in Figures 9.6a and 9.6b. With an increasing number of CBSs the execution time also increases and there is only a small influence from different capacity demands d. Overall, the heuristic algorithm is about four orders of magnitude

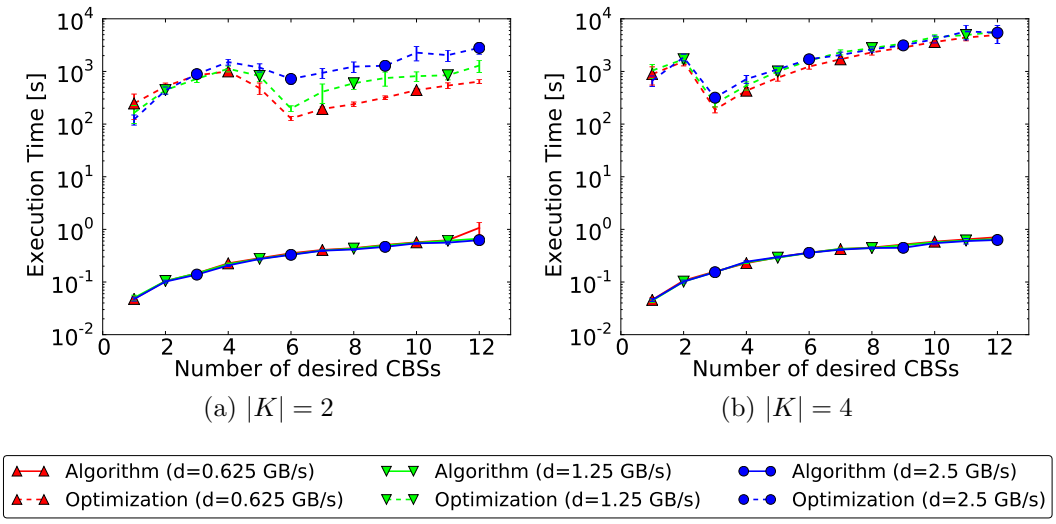


Figure 9.6: Execution Time

faster than solving the optimization problem.

This significant speedup and the overall closeness of agreement in the solution quality between the optimization problem and the heuristic algorithm make the heuristic algorithm suitable for efficient CBS selection and wavelength assignment during operation of the network.

9.4.3 Heuristic Algorithm in Large Scenarios

To evaluate the benefits of dynamic backhaul configuration by using our heuristic algorithm for controller selection and wavelength assignment, I use a scenario with 36 BSs. For the backhaul topology I both evaluate a mesh and a tree with a splitting factor of 9, which results in 4 central nodes. The number of CBSs is varied between 1 and 61 CBSs. Again I show evaluation results for 2 (solid lines) and 4 (dashed lines) available wavelengths per link. I call the case with 2 wavelengths the *baseline* case without reconfiguration and the case with 4 wavelengths the *reconfigurable* case.

The results for the number of feasible CBSs in Figures 9.7a and 9.7b show the expected behavior: with 4 available wavelengths in the *reconfigurable* case, more CBSs are feasible and can be established. I can also identify three tipping points at around 3, 7 and 16 CBSs for the tree topology and 2, 10, 20 for the mesh topology, respectively for $d = 2.5$, $d = 1.25$ and $d = 0.625$. At these tipping points the capacity with 2 available wavelengths is not sufficient to establish all desired CBSs and the feasibility starts to drop below one. These tipping points are marked in the plots by vertical dashed lines.

In the results for the used total wavelengths (Figure 9.8a) and the used wavelengths per link (Figures 9.9a and 9.9b) before the respective tipping points the number of used wavelengths is identical for both 2 and 4 available wavelengths. For

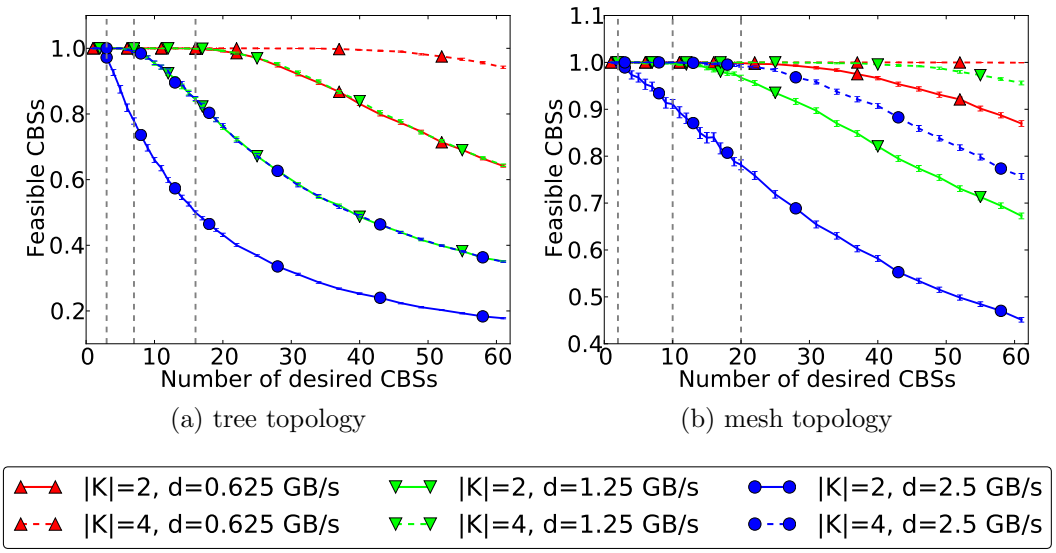
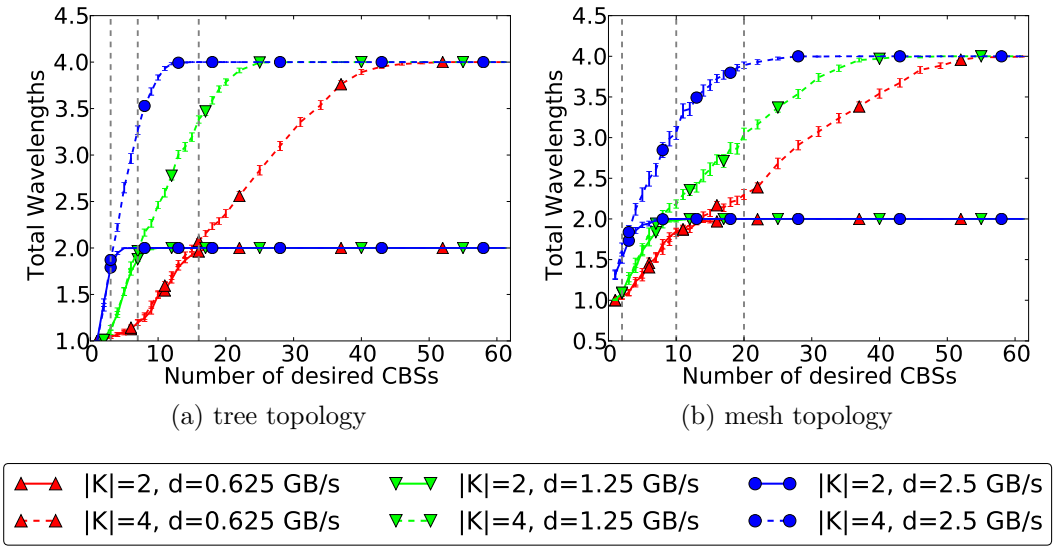


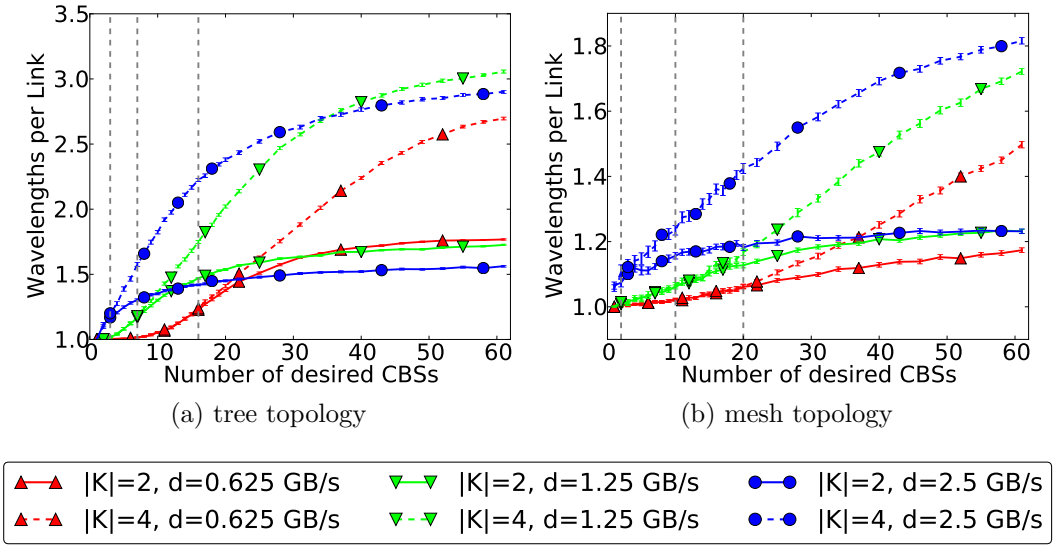
Figure 9.7: Feasible CBSs

the total number of used wavelengths in the mesh scenario (Figure 9.8b) the number of used wavelengths is identical until closely before the tipping points. This indicates that the algorithm does not assign additional available wavelengths until the traffic demands have increased to a level where not assigning additional wavelengths would impair the feasibility of BSs coordination, i.e. reduce the number of feasible CBSs. Also, after the tipping points the number of assigned wavelengths increases gradually and does not escalate immediately. This shows that the algorithm dynamically assigns wavelengths precisely according to the current demand.



(a) tree topology (b) mesh topology

Figure 9.8: Total Wavelengths

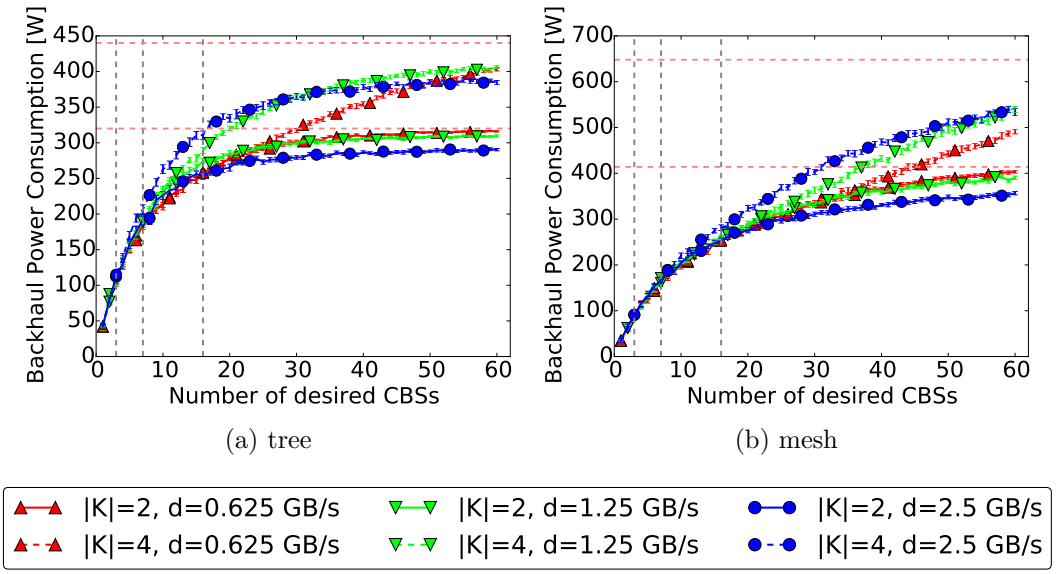


(a) tree topology (b) mesh topology

Figure 9.9: Wavelengths per Link

9.4.4 Energy Efficiency

Based on the power consumption model by Grobe et al. [GRA+11] which I have introduced in Section 2.1.2, it is possible to calculate the power consumption of the backhaul network with different assignments of wavelengths. The results for both the tree and mesh topologies are shown in Figures 9.10a and 9.10b and the reference power consumption of a full, static assignment of all wavelengths are indicated by horizontal, dashed lines in the plots, the lower one for $|K| = 2$ and the higher one for $|K| = 4$. As the number of desired CBSs increases, the power consumption of the backhaul network increases too. Especially in the tree topology the power consumption exhibits a logarithmic slope. Thus backhaul network reconfiguration can significantly reduce the power consumption during idle phases of the wireless access network. Even at the peak number of desired CBSs in this evaluation there is a notable gap to the reference power consumption.



(a) tree

(b) mesh

Figure 9.10: Backhaul Power Consumption

9.5 Summary

In this chapter, I presented an approach to verify the feasibility of Coordinated Base Station Sets (CBSs) with respect to backhaul network capabilities. My approach includes the dynamic assignment of backhaul resources (i.e., wavelengths and links). This new approach ensures that BS coordination mechanisms that are required for an optimal operation of the wireless part of the network are feasible in backhaul networks with limited resources.

My evaluation shows that the presented heuristic algorithm assigns backhaul resources only if they are needed. This is not possible with the model and algorithm in previous work (Section 8.1).

Considering the possible reduction of energy consumption by switching off unneeded WDM-PON equipment, my approach is also able to reduce the overall energy consumption of the backhaul network. Thus, my approach can contribute to shape the energy consumption of whole wireless access network towards a more traffic-proportional way.

10

Backhaul Network Reconfiguration Extension for DenseNets

The approach for backhaul network reconfiguration presented in the previous chapter combines the identification of feasible Coordinated Base Station Sets (CBSs) and the dynamic assignment of backhaul resources into a single step; this was separated in two different steps in previous work. In the evaluation in the previous chapter I focused on the comparison of the heuristic BFS Algorithm with an optimal solution and the performance of the BFS algorithm in large scenarios. This version of the BFS algorithm and the evaluation did not account for the densification of mobile access networks.

In this chapter I introduce an extension to the BFS algorithm to prioritize CBSs without external configuration. CBS prioritization enables the BFS algorithm to also deal with dense wireless access networks (DenseNets), where hotspots with a high density of users occur. I evaluate the extended BFS algorithm in scenarios with different densities of hotspots.

10.1 Problem Description

Regardless of the presence of hotspots of CBSs, the core decisions for each desired CBS remain the same as in Chapter 9:

1. Which node should be the controller for the CBS?

2. Which links and wavelengths in the backhaul should be used to form routing paths to exchange data between the controller and the Base Stations (BSs) in the CBS?

But an additional question as a consequence of the presence of hotspots of CBSs is:

3. To which CBSs should backhaul resources be assigned first, to keep the number of feasible CBSs high, especially in the hotspots?

In the next section I explain how the algorithm from Section 9.3 can be extended by an initial classification step to identify CBSs in the hotspots.

10.2 Hotspot BFS Algorithm

To deploy the algorithm in dense scenarios with hotspots of CBSs, I extend the heuristic algorithm from Section 9.3 by a CBS prioritization mechanism. This mechanism is explained in detail in this section together with a brief recap of the overall algorithm from Chapter 9 together with the extension.

The inputs for the heuristic are the same as in Chapter 9: the backhaul network as an annotated graph $G = (V, E)$, a set of available wavelengths per link K, and the desired CBSs W with each $W_i \subset V$ together with their constraints on data rate and latency.

An overview of the heuristic is depicted in Figure 10.1 and the individual steps are explained below. Steps 1 to 5 are the same as in the original BFS algorithm in Chapter 9 and are repeated briefly for completeness.

CBS Prioritization This new step of the algorithm divides the desired CBS into two sets $\mathrm{W}_{\mathrm{hotspot}}$ and $\mathrm{W}_{\mathrm{normal}}$. The calculations in this step use two threshold values to divide the desired CBSs: t_v, which determines how strict the filtering of hotspot vertices should be, and t_h, which determines how strict the filtering of hotspot CBS should be. The threshold values depend on the scenario and can be determined using a parameter study.

The algorithm first calculates for each vertex v from the backhaul graph in how many CBS the vertex is present as:

$$h_v = \sum_{W_i \in W} \mathbb{1}_{v \in W_i}$$

CBS Prioritization
▷ decide for each CBS if it belongs to a hotspot
▷ *output*: list of hotspot CBSs and normal CBSs
For each CBS in $\mathbf{W_{hotspot}}$, then for each CBS in $\mathbf{W_{normal}}$

1. **Maximum-Path BFS**
   ▷ start modified BFS from each vertex
   ▷ *output*: BFS tree for each vertex
2. **Match CBS**
   ▷ match BFS trees against CBS
   ▷ *output*: possible candidate BFS trees for CBS
3. **Back-Track BFS Trees**
   a) **Check constraints**
      ▷ recheck constraints on candidate BFS trees
   b) **Wavelength Assignment**
      ▷ determine wavelengths for all candidate BFS trees
   ▷ *output*: possible candidate BFS trees and their wavelength assignment for CBS
4. **Match CBS**
   ▷ match BFS trees against CBS again
   ▷ *output*: confirmed candidate BFS trees for CBS
5. **Find Best BFS Tree**
   ▷ compare candidate BFS trees
   ▷ *output*: best BFS tree for CBS

Figure 10.1: Hotspot BFS Algorithm

$\mathbb{1}_X$ denotes an indicator variable with value 1 if condition X is true and value 0 otherwise. A given vertex v is considered as a hotspot vertex if

$$h_v > |W| \cdot t_v$$

and is added to the set of hotspot vertices V_h. Now each CBS W_i is added to $\mathrm{W}_{\mathrm{hotspot}}$ if

$$\frac{\sum_{v \in W_i} \mathbb{1}_{v \in V_h}}{|W_i|} \geq t_h$$

otherwise it is added to $\mathrm{W}_{\mathrm{normal}}$.

Now the actual algorithm for placement of the controller and backhaul resource allocation has to be executed for each CBS. The following steps are executed per CBS, starting with the CBSs from $\mathrm{W}_{\mathrm{hotspot}}$ and then the CBSs from $\mathrm{W}_{\mathrm{normal}}$.

The algorithm steps for the placement of the controller and backhaul resource allocation for each CBS are identical to the algorithm described in Section 9.3.

10.3 Evaluation

In this section I present simulation results that show how the extended algorithm performs compared to the original algorithm from Chapter 9.

10.3.1 Hotspot Scenario

Consistent with the evaluation in Chapter 9, a fixed number of BSs are placed on a regular grid, with a mean inter-BS distance of $\bar{s} = 1000\,\text{m}$ (urban scenario [BAWB13]), and are then shifted in both x and y direction according to two independent, normally distributed random variables with zero mean and standard deviation $\frac{\bar{s}}{8}$.

The backhaul network is created as in Chapter 9 with two different topologies:

1. As a mesh topology where two BSs are connected by a link if their distance is less than or equal to $1.5 \cdot \bar{s}$.

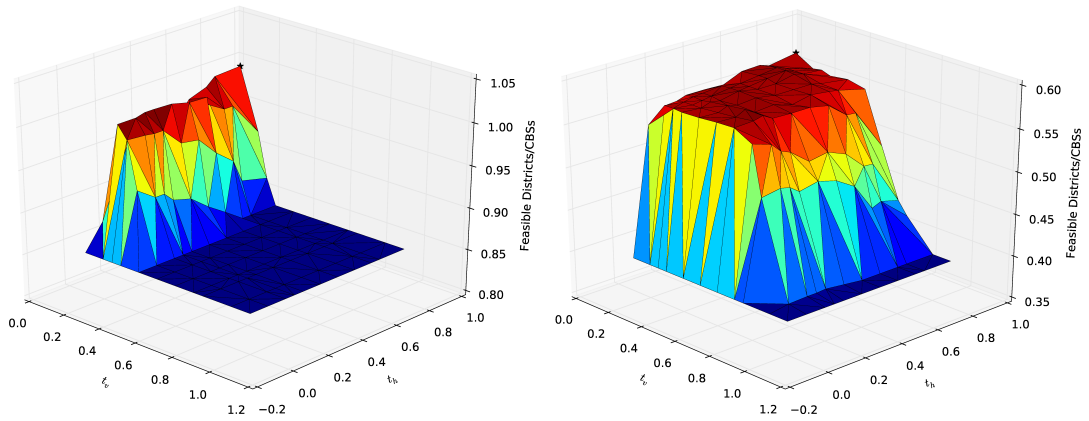
2. As a tree topology where the BSs in the same area are connected to a common central node according to a splitting factor. All central nodes are located in a central site and are fully interconnected.

All links in the backhaul network are assigned the same set of available wavelengths $K = 4$ and each wavelength is assigned the same fixed capacity of 2.5 Gb/s. The latency for each link is again determined by the distance multiplied by 1.45 divided by the speed of light as I am modeling an optical backhaul network.

In order to create a hotspot of CBSs, an (x,y) coordinate is selected uniformly at random as the hotspot center. Now a fraction h of all CBSs is placed around the hotspot center based on a normal distributions for x and y coordinates with the hotspot coordinate as the mean and standard deviation $\frac{\bar{s}}{4}$ as the desired hotspot CBSs. All other desired CBSs are generated by placing them uniformly at random on the plane covered by the placed BSs. The BSs that are considered as the CBS are all BSs located inside a circle around the coordinates of the CBSs with a given radius. I determine this radius by multiplying the mean inter-BS distance with a factor $r = 1.5$, which results in 5 BSs per CBS on average.

The capacity demand d for each BS in the CBSs is set to the same value and is either 0.625 Gb/s, 1.25 Gb/s, or 2.5 Gb/s. This implies that at a demand of 2.5 Gb/s one complete wavelength is required to connect a BS to the controller.

To determine the threshold values for the CBS prioritization t_v and t_h, I performed a parameter study and identified $t_v = 0.1$ and $t_h = 0.9$ as the best values to maximize the number of feasible CBSs in this scenario. Figure 10.2 shows two graphical example outputs from the parameter study for $d = 1.25\,\text{Gb/s}$ and 45 desired CBSs.



(a) $h = 0$ (b) $h = 0.75$

Figure 10.2: Parameter study examples

10.3.2 Hotspot Scenario Results

The results for the simulation are shown in Figures 10.3 and 10.4, where I investigate the influence of different hotspot CBS fractions h for both the mesh and tree topologies. Each row of plots shows the results for a value of h written above the row, each column of plots consists of plots with same metric. For $h = 0$ no hotspot CBSs exist and for $h = 1$ all CBSs are hotspot CBSs. Solid lines correspond to the extended algorithm with the CBS prioritization step, dashed lines the algorithm from Section 9.3 as a reference.

In Figures 10.3a and 10.4a I show the resulting feasibility of the CBS, i.e., the fraction of desired CBSs that were successfully established. Even for a scenario with no hotspots ($h = 0$) the CBS prioritization step increases the CBS feasibility to 1 for all capacity demands and all numbers of desired CBSs in the mesh topology. In the tree topology the prioritization step increases the CBS feasibility to between 0.95 and 1 compared to worst-case values between 0.4 and 0.6 without prioritization. This indicates that the prioritization step is also beneficial for scenarios without explicit hotspots of CBSs. In the scenarios with $h = 0.5$ and $h = 0.75$, the feasibility drops as the number of desired CBSs increases for both topologies. For both $h = 0.5$ and $h = 0.75$ the CBS prioritization step increases the CBS feasibility by around 0.2 with a slightly larger increase in the scenarios with $h = 0.5$. This shows that the CBS prioritization step is necessary for scenarios with hotspots of CBSs. In the scenario with all desired CBSs in the hotspot ($h = 1$), there is now significant improvement from prioritizing CBSs, which is the expected outcome as both algorithms work the same in such a scenario.

Figures 10.3b and 10.4b show the results for the overall number of used wavelengths and Figures 10.3c and 10.4c show the results for the number of used wavelengths per link. In the scenario with no hotspot CBSs ($h = 0$) the CBS prioritization significantly improves the efficient use of wavelengths, as both the total number of used wavelengths and the number of wavelengths per link are significantly lower when using the CBS prioritization. This indicates that assign-

ing backhaul network resources with a prioritization step in a scenario without explicit hotspots of CBSs is also beneficial for reducing the number of used backhaul network resources. This effect is also evident in the scenarios with $h = 0.5$ and $h = 0.75$, but the difference to the algorithm without the prioritization step is smaller compared to the scenario with $h = 0$. This shows that the presence of hotspots of CBSs significantly increases the demand of required backhaul network resources and that the enhanced algorithm is capable to successfully satisfy this demand considering the results for wavelength assignment and CBS feasibility together. For $h = 1$ there is no significant difference between the two algorithm, which again is the expected behavior.

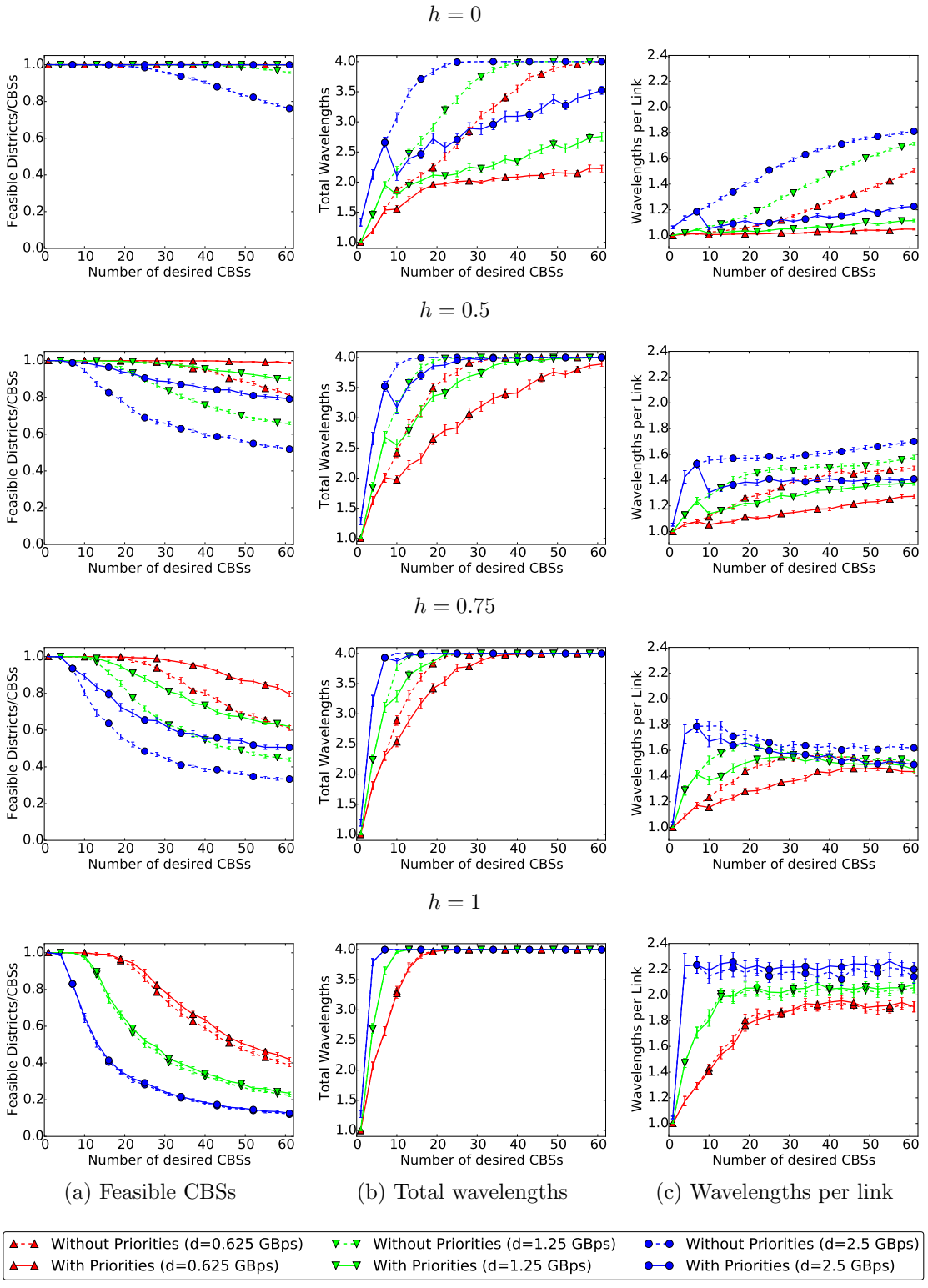


Figure 10.3: CBS Prioritization Simulation (mesh)

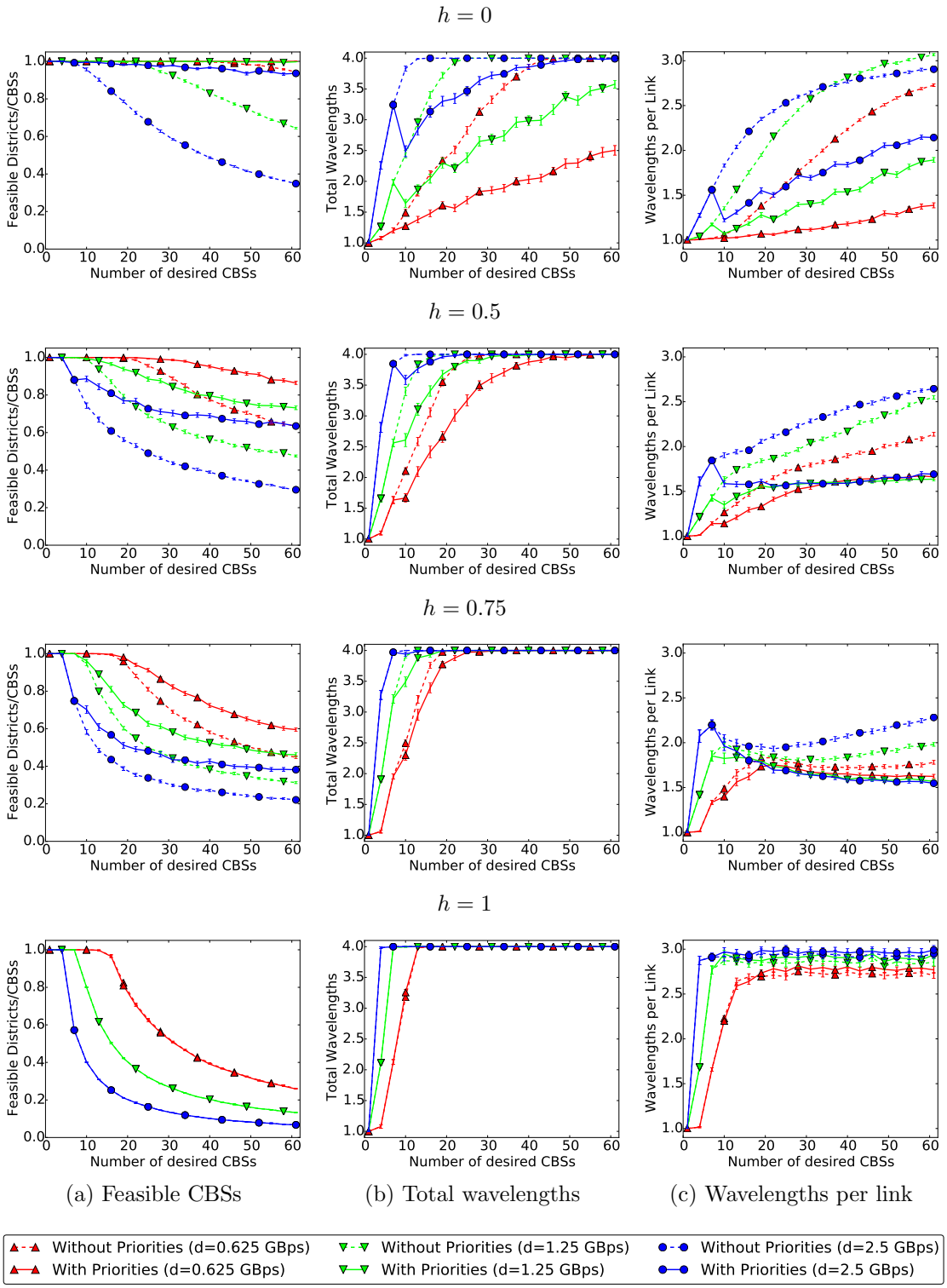


Figure 10.4: CBS Prioritization Simulation (tree)

10.3.3 Non-Hotspot Scenario Extended Results

Because of the indicated improved performance of the algorithm in non-hotspot scenarios, I investigated these scenarios further and performed the same evaluation as in Section 9.4.3 where I evaluated the performance of the BFS algorithm in large scenarios. Again I use a tree and a mesh topology scenario, each with 36 BSs and show results for 2 (solid lines) and 4 (dashed lines) available wavelengths per link. In the previous results I had shown how the BFS algorithm without CBS prioritization efficiently assigns wavelengths in the *reconfigurable case* with 4 available wavelengths compared to the *baseline case* with 2 available wavelengths. Here I want to evaluate the influence of the added CBS prioritization.

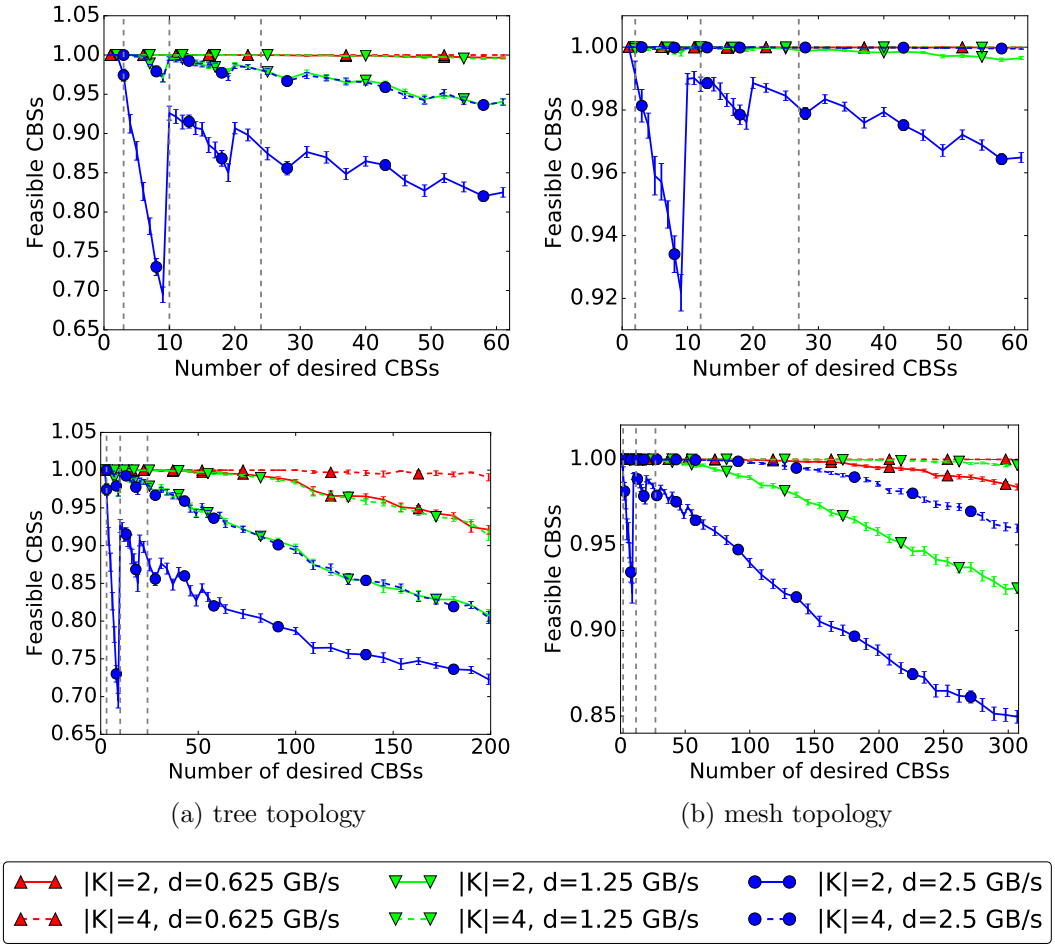
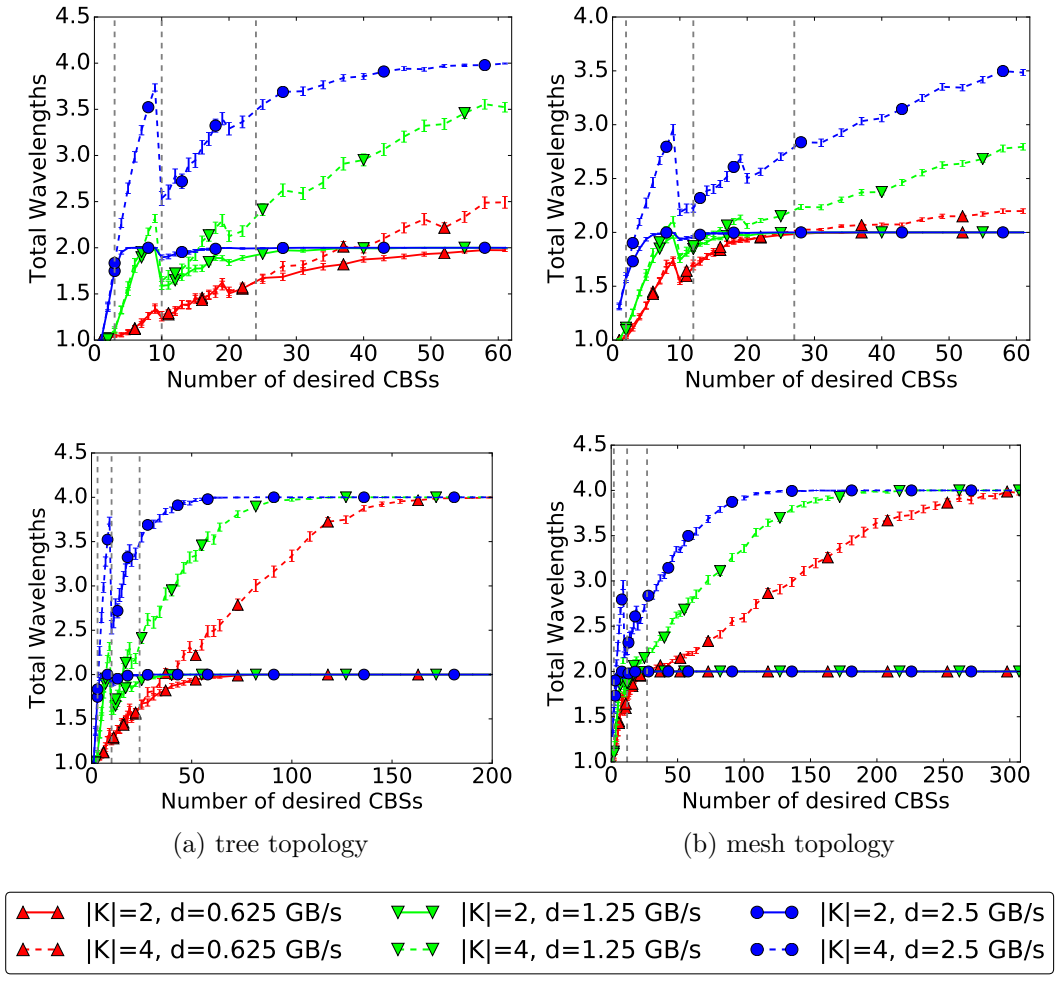


Figure 10.5: Feasible CBSs

Figures 10.5a and 10.5b show the new results for the number of feasible CBSs: in the upper plot zoomed in to 1 to 61 desired CBSs as in the previous evaluation and in the lower plot for the full range of up to 200 and 300 desired CBSs respectively for the tree and mesh topology. There are two major differences to the previous evaluation: First, especially when looking at a range of 1 to 61 desired

CBSs, the lines are less smooth with a notable jump around 9 desired CBSs, especially for $d = 2.5$. This effect is a result of the CBS prioritization step because the prioritization of CBSs always happens with respect to the total set of CBSs. With a small set of total CBSs without explicit hotspot CBSs the algorithm does not always prioritize CBSs in the best way and the CBS prioritization provides no advantage yet. Second, and more importantly, the number of feasible CBSs is significantly improved compared to the previous evaluation without CBS prioritization. Without CBS prioritization the number of feasible CBSs started to decrease below 1 for $d = 0.625$ around 60 desired CBSs. With CBS prioritization, this decrease occurs at around 200 desired CBSs in the tree topology and at around 300 desired CBSs in the mesh topology. For $d = 1.25$ and $d = 2.5$ there is a similar improvement and CBS prioritization improves the number of feasible CBSs by a factor of around 3 in the tree topology and 5 in the mesh topology.

The plots again contain vertical dashed lines indicating tipping points where the algorithm starts to assign more than 2 wavelengths if there are $|K| = 4$ wavelengths available per link.

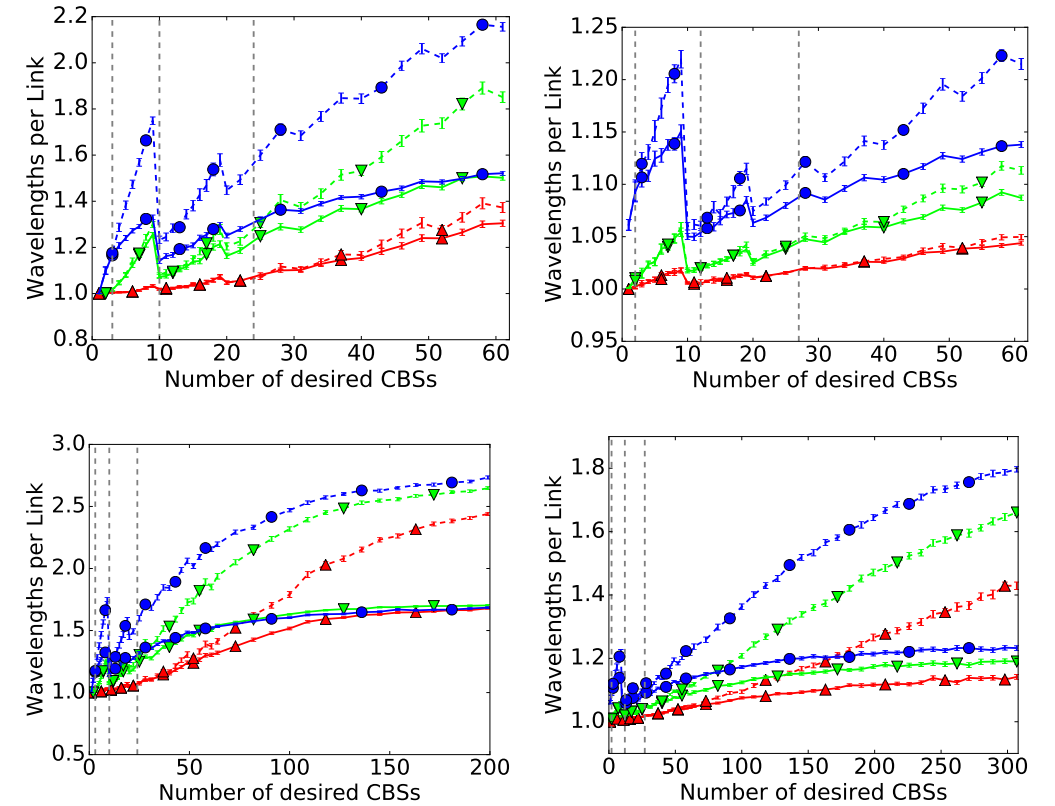


(a) tree topology (b) mesh topology

Figure 10.6: Total Wavelengths

The results for the used total wavelengths are shown in Figures 10.6a and 10.6b and the results for the used wavelengths per link in Figures 10.7a and 10.7b. The tipping points (vertical dashed lines) also identified in the previous evaluation exist again at around 3, 10 and 24 CBSs for the tree topology and 2, 12, 27 for the mesh topology, respectively for $d = 2.5$, $d = 1.25$ and $d = 0.625$. At these tipping points the capacity with $|K| = 2$ available wavelengths is not sufficient to establish all desired CBSs and the algorithm starts to assign more than 2 wavelengths.

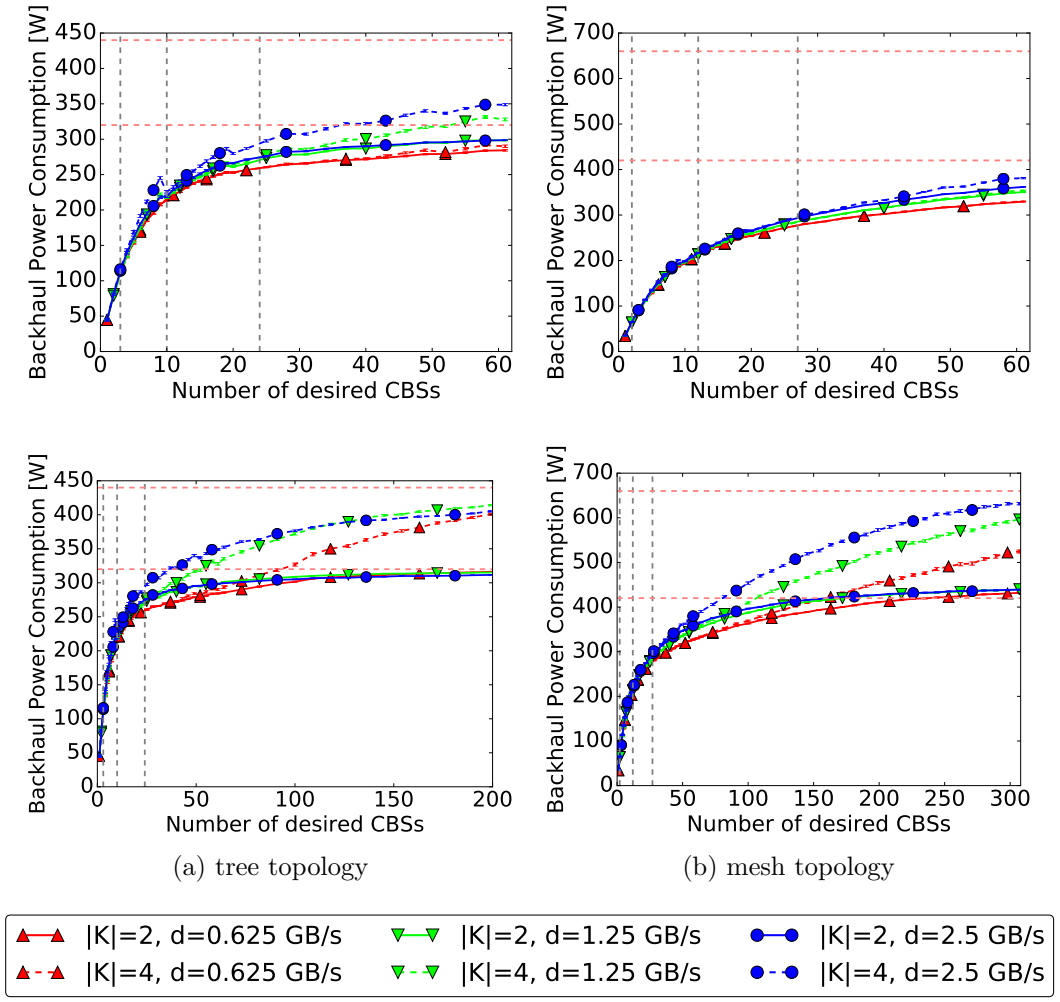
The notable jump at around 9 desired CBSs is also visible in the results here for all values of d and is again due to the CBS prioritization not working advantageously with a small total set of CBSs. Apart from that, the results are consistent with the previous results: the algorithm does not assign additional available wavelengths until the capacity demands have increased so much that additional wavelengths are required to maintain the feasibility of BSs coordination, i.e., keep the number of feasible CBSs at 1. Again, after the tipping points wavelength are assigned gradually.



(a) tree topology (b) mesh topology

Figure 10.7: Wavelengths per Link

Figures 10.8a and 10.8b show the backhaul power consumption based on the power consumption model by Grobe et al. [GRA+11] (see Section 2.1.2) as a comparison to the previous results in Section 9.4.4. Again the reference power consumption of a full, static assignment of all wavelengths are indicated by horizontal, dashed lines, the lower one for $|K| = 2$ and the higher one for $|K| = 4$. The results for the backhaul power consumption are consistent with the other results: with CBS prioritization the wavelengths are assigned more efficiently and thus the power consumption in the range of 1 to 61 desired CBSs is lower compared to the previous results without CBS prioritization. For the full range of 1 to 200 and 300 desired CBSs respectively, the slope and final value of the backhaul power consumption is similar to the previous results without CBS prioritization. Thus, CBS prioritization does not have a negative influence on the energy consumption of my approach for backhaul network reconfiguration.



(a) tree topology (b) mesh topology

Figure 10.8: Backhaul Power Consumption

10.4 Summary

In this chapter, I have presented an extension to my approach to verify the feasibility of base station coordination considering capacity and latency constraints with a backhaul network with limited resources and hotspots of CBSs.

My simulations show that the extension enables the use of my approach in *dense* wireless access networks with hotspots of users and furthermore significantly increases the feasibility of base station coordination in scenarios without hotspots.

11

Backhaul Network Reconfiguration Prototype

Based on the CROWD Controller Architecture (CCA) (Section 8.2) and the Software-Defined Network (SDN) paradigm (Section 2.1.3) I have designed and implemented a prototype for backhaul network reconfiguration.

My approach for backhaul network reconfiguration needs to be executed on a regional scope according to the CCA and is thus implemented as an application for the CROWD Regional Controller (CRC) based on OpenDayLight (ODL). In line with the ideas behind the CCA the backhaul is implemented using OpenFlow.

11.1 Architecture

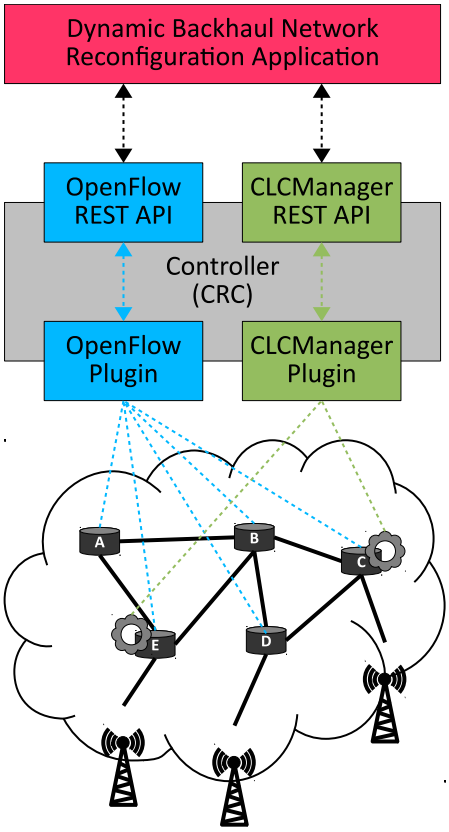


Figure 11.1: System Architecture

The architecture of the prototype is based on the decoupling of the following three components of the overall system as shown in Figure 11.1 as a subset of the CCA (Section 8.2):

- Application
- Controller (CRC)
- Backhaul network

In my prototype setup, the application is the algorithm from Section 10.2 together with necessary data structures to store the input and output data at runtime. The system architecture itself can accommodate any other application for backhaul reconfiguration.

The application is decoupled from the controller by accessing two different Application Programming Interfaces (APIs) of the controller. Via the OpenFlow API the application can both query the topology and flow configuration of the backhaul network and modify the flow configuration according to the algorithm outputs. In addition to this API, the application also needs to access

the CLC Manager API in order to query the status of running CROWD Local Controllers (CLCs) and to start or stop CLCs.

The CROWD Regional Controller (CRC) acts as the centralized link between the application and the backhaul network, like an SDN controller [ONF12]. It exposes the previously described northbound APIs to the application and controls both the backhaul network and the CLC instances via its southbound plugins. Consistent with the northbound APIs the controller needs both an OpenFlow plugin and a CLC Manager plugin.

Of course the backhaul network has to be based on OpenFlow-enabled hardware, i.e., switches, otherwise the controller cannot use the OpenFlow plugin to reconfigure the backhaul network. All potential nodes for hosting a CLC have to run a hypervisor to allow the dynamic instantiation of CLC instances.

11.2 Implementation

My reference implementation of the described system architecture is based on the OpenDayLight (ODL) controller platform [MTVG14, ODL]. ODL includes a fully featured OpenFlow plugin, thus no additional implementation is required for this part. The model-based design behind ODL facilitates the implementation of a CLC Manager plugin. A core concept with ODL is the implementation of REST-based northbound APIs. The OpenFlow plugin provides APIs for both querying the topology and configuring flows within the backhaul network. For the CLC control plugin, I have implemented REST APIs accordingly. I describe this implementation in more detail in Section 11.2.1.

The implementation of the application is based on Python and solely relies on the REST APIs of the controller. A small wrapper converts data between the ODL API format and the data structures for the algorithm. This wrapper is tailored to the ODL API format, but it can be adapted to any other OpenFlow controller, making the algorithm implementation independent of the controller platform.

Because I cannot test the application and controller on a real-world OpenFlow-enabled backhaul network, I have used an emulated network to test and evaluate my implementation. For this I use Maxinet [WDS+14], an extension to the well known Mininet emulator [LHM10] for distributed emulation, to emulate a fully functional, virtual OpenFlow-enabled network on a cluster of physical machines. I describe Maxinet in more detail in Section 11.2.2.

11.2.1 Controller and CLC Manager Plugin

Based on the two-tier architecture of the CCA (Section 8.2) the backhaul network reconfiguration should be implemented as a regional control mechanism to be executed on the CRC. Because OpenDayLight (ODL) is the target platform for the CRC I had to extend ODL with the necessary functionality for backhaul network reconfiguration.

The architecture of ODL is shown in Figure 11.2. With the existing OpenFlow plugin, the CRC controls OpenFlow-enabled devices. To allow access to the APIs

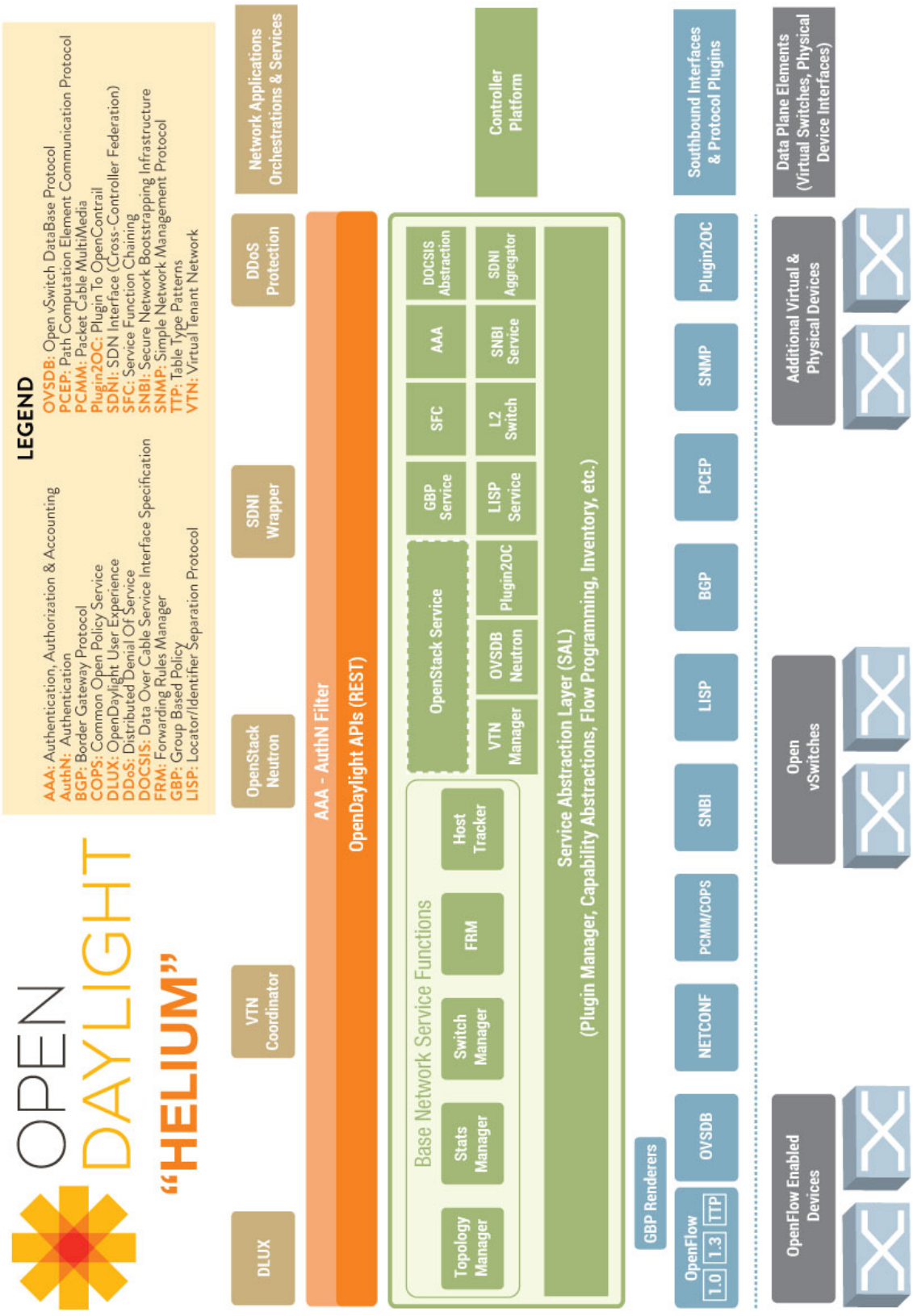


Figure 11.2: OpenDaylight architecture [ODL]

of the CLCs controlled by the CRC and to control the CLC instances, I have developed a new set of modules for ODL called *CLCManager*.

The modules of the *CLCManager* are shown in Figure 11.3 and described in the following:

- CLCManager Interface: provides a northbound Representational State Transfer (REST) API and translates queries between this API and the core modules.
- CLCManager Service: handles commands and function calls to the core plugin according to the Service Abstraction Layer (SAL) architecture of ODL.
- CLCManager Plugin: manages the CLC inventory
- Inventory: data structure for the actual CLC inventory.
- CLCConnector: handles the communication with the CLCs (Because the prototype does not include real instances of the CLC, this module is only implemented as a non-functional stub)

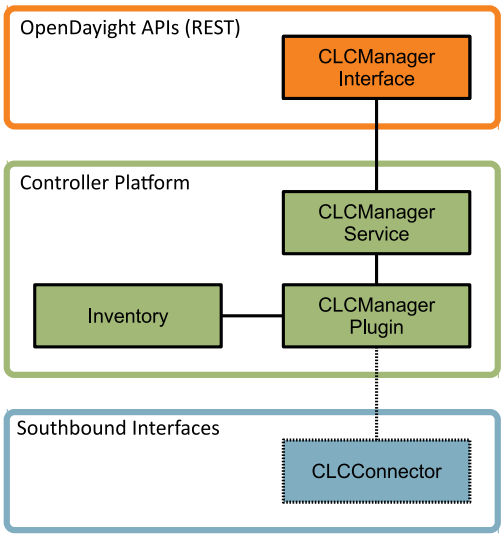


Figure 11.3: CRC modules in ODL

11.2.2 Backhaul Network with Maxinet

Maxinet is an abstraction layer to connect multiple, *unmodified* instances of Mininet [LHM10] running on different physical hosts. It provides a centralized API for accessing this cluster of Mininet instances and uses GRE tunnels [Dom00] to interconnect virtual switches emulated on different physical hosts. Maxinet works as a frontend for Mininet that sets up all Mininet instances, invokes commands at the virtual nodes and sets up the tunnels required for proper connectivity.

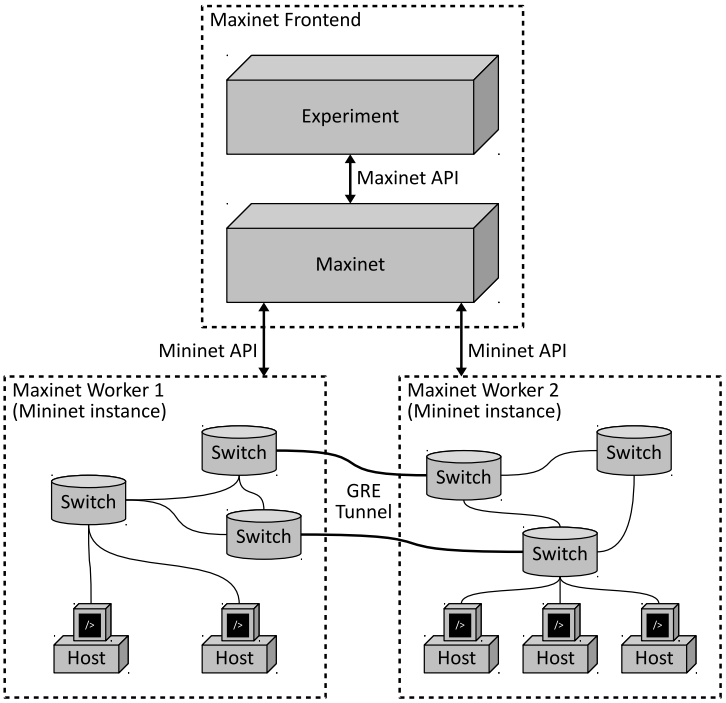


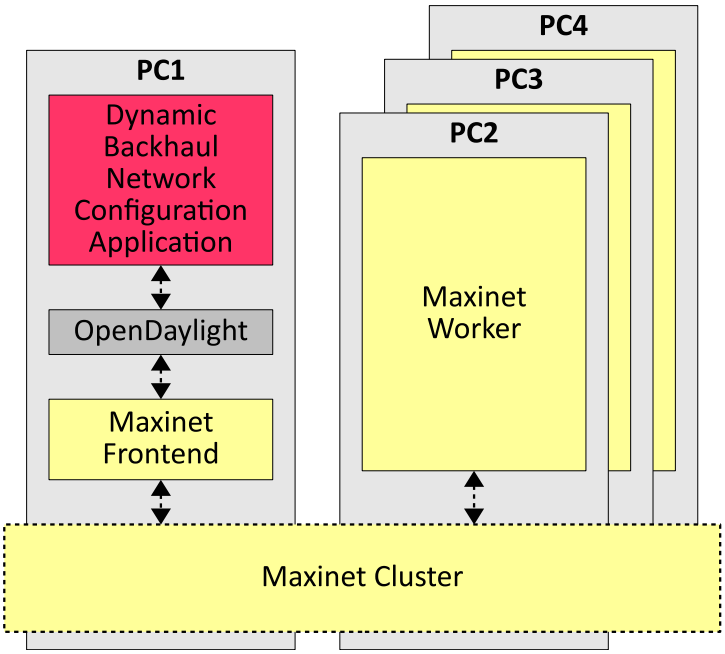
Figure 11.4: Schematic view of Maxinet

Figure 11.4 shows a schematic view of Maxinet. A network experiment can use the Maxinet API to set up, control and shut down a virtual network. This Maxinet API is designed to be very close to the Mininet API to easily distribute existing experiments over a cluster of Mininet instances. The emulation of the network as such happens on a cluster of workers. All Workers are controlled by Maxinet using the Mininet API. Communication between Maxinet and Mininet is implemented through RPC calls.

I developed Maxinet together with Philip Wette, Arne Schwabe, Felix Wallaschek and Mohammad Hassan Zahraee [WDS+14].

11.2.3 Prototype Setup

I use a small setup with four physical machines, as shown in Figure 11.5. One machine hosts the backhaul network reconfiguration , OpenDaylight, and the Maxinet frontend; the three other machines are used as Maxinet workers to emulate the backhaul network. Since I do not have any wireless interface integrated into the testbed, the traffic between the Base Stations (BSs) and the CLCs is emulated using iperf [HK98].



(a) Testbed Architecture

(b) Testbed Setup

Figure 11.5: Testbed

11.3 Evaluation

For evaluating the prototype I used my prototype described in Section 11.2.

11.3.1 Scenario

For the prototype evaluation I use a smaller scenario, compared to the simulation scenario, with only 16 BSs. To avoid a bottleneck from the Maxinet worker interconnect, which is provided by a 100 Mb/s Ethernet switch, a Coordinated Base Station Set (CBS) can only contain BSs being emulated on the same worker machine. Apart from this limitation, CBSs are generated in the same random way as in the simulations (Section 9.4). All emulated links also have a capacity of 2.5 Gb/s, and for the demand per BS I only consider 1.25 Gb/s because the other values from the simulation (0.625 Gb/s and 2.5 Gb/s) do not provide additional insights from the evaluation results.

I also use three different implementations for the application to compare the performance of the algorithm:

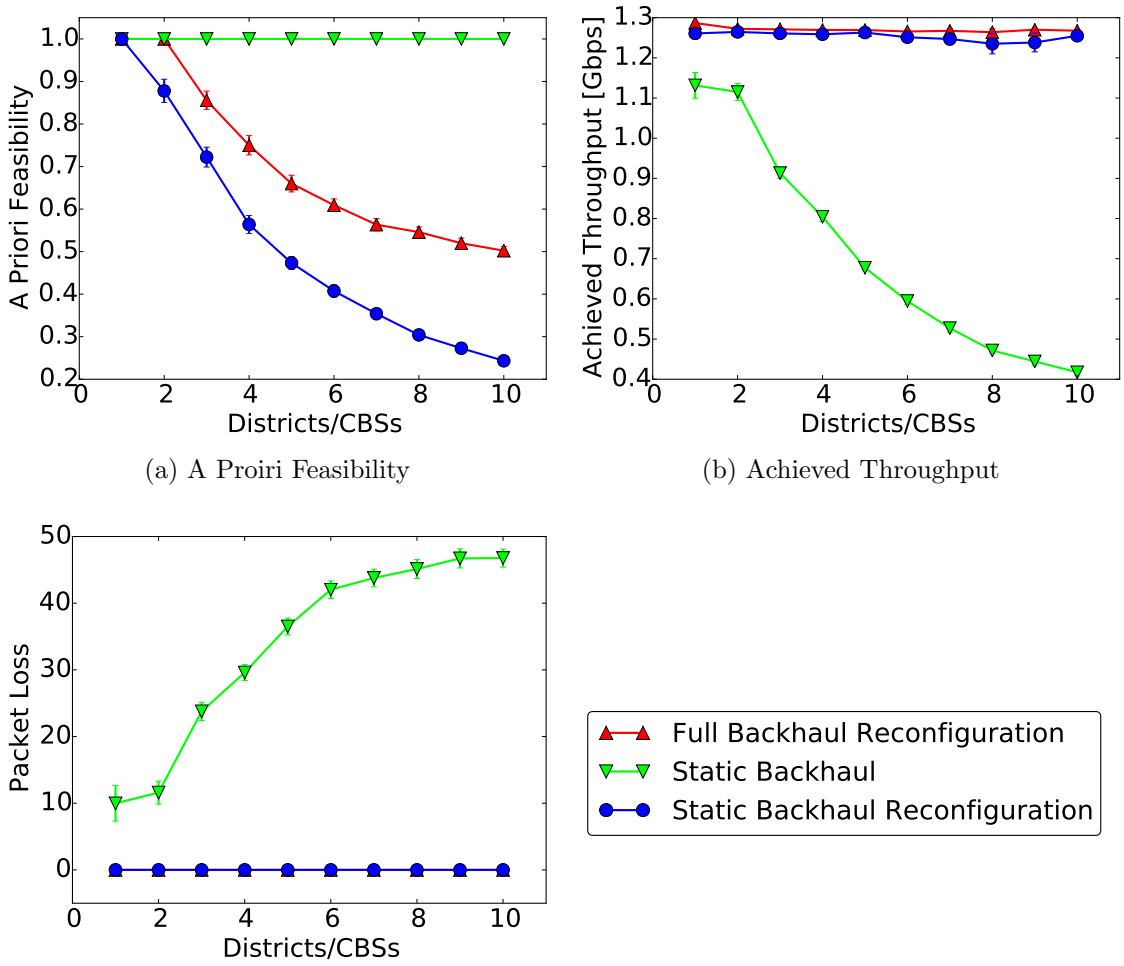
- *Full Backhaul Reconfiguration* uses the full algorithm described in Section 10.2 with the full flexibility in terms of backhaul network configuration and CLC placement.
- *Limited Backhaul Reconfiguration* uses a limited version of the algorithm where the CLC is placed on a fixed BS and the algorithm only performs flow routing and wavelength assignment.
- *Static Backhaul* does not use the algorithm at all and relies on a fixed CLC placement and a static assignment of wavelengths and always uses shortest paths for the flow routing.

11.3.2 Results

For the prototype evaluation, I consider three different metrics.

The *a priori feasibility* is calculated from the output of the application. In Figure 11.6a we can see that the *a priori feasibility* decreases for both applications that are based on the algorithm as the number of desired CBSs increases. This is due to the limited available resources in the backhaul network. The results also show that using the full flexibility of the algorithm yields a better *a priori feasibility* than using the limited version. The *a priori feasibility* for the static backhaul implementation is always 100% because this implementation does not consider any resource constraints in an a priori way.

In order to measure the *a posteri feasibility* of all implementations I perform a UDP throughput measurement using iperf [HK98] with a desired throughput of 1.25 Gb/s. As results from this measurement I obtain both the achieved throughput (Figure 11.6b) and the packet loss (Figure 11.6c). Both implementations with the algorithm achieve the desired throughput and do not cause any packet loss.



(a) A Proiri Feasibility

(b) Achieved Throughput

(c) Packet Loss

Figure 11.6: Prototype Evaluation

In contrast to that, the static implementation is not able to achieve the desired throughput and causes a significant packet loss.

11.4 Summary

In this chapter, I have presented a prototype implementation for backhaul network reconfiguration based on the CROWD Controller Architecture (CCA) (Section 8.2). With this prototype I have shown how backhaul network reconfiguration can be implemented as an application to run on top of the OpenDayLight SDN controller to dynamically reconfigure a backhaul network implemented with OpenFlow.

My evaluation results show that backhaul network reconfiguration improves the feasibility of BS coordination in such a real-world setup.

12

Conclusion & Future Research Directions

In Section 1.2, I have introduced four goals for the approaches I presented in this thesis. In Section 12.1 I discuss how my approaches contribute to achieving these goals. After that I conclude my work in Section 12.2 and outline future research directions in Section 12.3.

12.1 Discussion

In this section I discuss how my approaches contribute to achieving the four goals introduced in Section 1.2.

1. **Efficient use of wireless resources**
   My approach for *anticipatory download scheduling* directly contributes to this goal by preventing excessive buffering of video segments in situations where buffering of segments is not (yet) required. Wireless resources that are not wasted by such early downloads of segments can be used by other users, whose future available data rates might require the download and buffering of more segments.

   Backhaul network reconfiguration does not directly interact with the wireless resources but it enables wireless coordination in situations where the limitations of a static backhaul prevent the implementation of wireless coordination. Wireless coordination itself greatly enhances the efficient use of wireless resources and thus *backhaul network reconfiguration* indirectly contributes to achieving this goal as well.

2. **Efficient allocation of backhaul resources**
   As I have pointed out in my evaluations in Chapters 9 and 10, my ap-

proach for *backhaul network reconfiguration* allocates Wavelength-Division-Multiplexed Passive Optical Network (WDM-PON) wavelengths exactly where they are required by the coordination of Base Stations (BSs). I developed this approach with this goal in mind and thus it directly contributes to achieving it.

3. **Limited QoE degradation**
   I developed my approach for *anticipatory download scheduling* with the goal to decrease the degradation of the QoE for users as much as possible. My evaluation results in Chapters 4, 5 and 6 focus on the QoE for users with respect to playback interruptions (lateness) and video quality. My approach does not degrade the QoE but actually increases it significantly compared to greedy download scheduling. My approach greatly reduces the number of playback interruptions and especially the Plan algorithm (Chapter 5) delivers the video in a high quality to the users.

   Backhaul network reconfiguration has no direct implication on the QoE for users but it enables wireless coordination, which in turn helps to achieve a better QoE for the users.

4. **Reduced energy consumption**
   In Chapters 6 and 9, I have shown evaluations specifically addressing the influence of my approaches on the power consumption of the mobile access network.

   I have especially presented an extension for increased energy efficiency for the *anticipatory download scheduling* approach in Chapter 6. My results show the potential for a significant reduction of power consumption by using *anticipatory download scheduling* as an enabling approach to power off unused BSs.

   My evaluation results in Chapter 9 also show that the dynamic allocation of backhaul resources with *backhaul network reconfiguration* decrease the power consumption of the backhaul network, if unneeded backhaul equipment is powered off.

12.2 Conclusion

In all results that evaluate the behavior of state of the art approaches together with the increased traffic demands in future mobile access networks, I have seen substantial drawbacks in QoE and energy efficiency. In contrast to that, I have improved both QoE and energy efficiency with the new approaches I have proposed in this thesis. These results lend evidence to the fact that new, optimized approaches are needed to continuously operate mobile access networks in an efficient way. My approaches can help to achieve this goal.

12.3 Future Research Directions

For the *anticipatory download scheduling* approach, the following directions should be investigated further:

Integrated PHY scheduling The current anticipatory download scheduling approach is implemented in the application layer. In my investigation of energy efficiency I have also included the allocation of Physical Layer (PHY) resources but have not investigated it further, to keep the approach compatible with existing radio access protocols and standards. But combining download scheduling on the application layer with the scheduling and allocation of physical resources can increase the benefits of the approach even more.

Full multi-user scheduling After integrating PHY scheduling, it would also be possible to extend the approach to include the explicit swapping of resources between multiple users. This would allow multi-user fairness based on the delivered video quality and lateness instead of fairness only based on the allocation of PHY resources. Such an approach would be a full cross-layer approach combining fairness in both the PHY and application layer.

Prototype field test My prototype is only implemented as a stationary testbed with emulated user movement but all the components of the system are deployable in a larger field test with real moving users and a full implementation of a predictor for future available data rates.

Refined prediction error model The current prediction error model is based on the analysis of common prediction algorithms. After deploying a real field test with the prototype, it would be possible to derive a refined prediction error model that is tailored to the specific use case of mobile video streaming.

Furthermore, for *backhaul network reconfiguration* the following issues should be researched:

WDM-PON conversion model The current model for WDM-PONs in the optimization problem and the algorithms does not explicitly consider the conversion of wavelengths. If this model included a distinction between optical-optical and optical-electrical-optical conversion, it would be possible to derive even more precise results about energy efficiency and deployment costs.

Integrated prototype The prototype implementation does not include any wireless transmissions and wireless coordination. With proper wireless testbed equipment it would be possible to integrate and test the prototype implementation with a real wireless access networks to further show its real world applicability.

Acronyms

3GPP	3rd Generation Partnership Project
AP	Access Point
API	Application Programming Interface
ARMA	Autoregressive Moving Average
AWG	Array Waveguide Grating
BFS	Breadth-First-Search
BS	Base Station
CBS	Coordinated Base Station Set
CCA	CROWD Controller Architecture
CLC	CROWD Local Controller
CoMP	Coordinated MultiPoint transmission and reception
CRC	CROWD Regional Controller
CSI	Channel State Information
DMM	Distributed Mobility Management
DPI	Deep Packet Inspection
FSO	Free Space Optics
HLS	HTTP Live Streaming
HMM	Hidden Markov Model
ILP	Integer Linear Program
ISD	Inter Site Distance
LTE	Long Term Evolution
MDP	Markov Decision Process
MIMO	Multiple Input Multiple Output

MIQCP	Mixed Integer Quadratically Constrained Program
MME	Mobility Management Entity
ODL	OpenDayLight
OLT	Optical Line Terminal
ONF	Open Networking Foundation
ONU	Optical Network Unit
PDF	Probability Density Function
PHY	Physical Layer
PON	Passive Optical Network
PRB	Physical Resource Block
PTAS	Polynomial-Time Approximation Scheme
PtP	Point-to-Point
QoE	Quality of Experience
RB	Resoure Block
REST	Representational State Transfer
RSSI	Received Signal Strength Indication
SAL	Service Abstraction Layer
SDN	Software-Defined Network
SINR	Signal to Interference and Noise Ratio
SPAN	Smarter Phones And Networks
SVC	Scalable Video Coding
TRX	Transmission and Reception Unit
UE	User Equipment
URL	Uniform Resource Locator
WDM	Wavelength Division Multiplexing
WDM-PON	Wavelength-Division-Multiplexed Passive Optical Network

Bibliography

[3GP09] 3GPP. Further advancements for E-UTRA physical layer aspects. Technical Report 36.814 V9.0.0, 3GPP, March 2009.

[AACdlO+13a] H. Ali-Ahmad, C. Cicconetti, A. de la Oliva, M. Dräxler, R. Gupta, V. Mancuso, L. Roullet, and V. Sciancalepore. CROWD: An SDN Approach for DenseNets. In *Second European Workshop on Software Defined Networks*, 2013.

[AACdlO+13b] H. Ali-Ahmad, C. Cicconetti, A. de la Oliva, V. Mancuso, M. R. Sama, P. Seite, and S. Shanmugalingam. An SDN-based Network Architecture for ExtremelyDense Wireless Networks. In *Proceedings of IEEE Software Defined Networks for Future Networks and Services (IEEE SDN4FNS)*, 2013.

[ADMM15] S. Auroux, M. Dräxler, A. Morelli, and V. Mancuso. Dynamic network reconfiguration in wireless DenseNets with the CROWD SDN architecture. In *Proceedings of the 2015 European Conference on Networks and Communications (EuCNC)*, 2015.

[And13] J. G. Andrews. Seven ways that HetNets are a cellular paradigm shift. *IEEE Commun. Mag.*, 51(3):136–144, 2013.

[App] Apple Inc. HTTP Live Streaming. `https://developer.apple.com/resources/http-streaming`.

[ASM15] A. Asadi, V. Sciancalepore, and V. Mancuso. On the efficient utilization of radio resources in extremely dense wireless networks. *IEEE Communications Magazine*, 53(1):126–132, January 2015.

[AzHV14] H. Abou-zeid, H.S. Hassanein, and S. Valentin. Energy-efficient adaptive video transmission: Exploiting rate predictions in wireless networks. *Vehicular Technology, IEEE Transactions on*, 63(5):2013–2026, Jun 2014.

[BAWB13] O. Blume, A. Ambrosy, M. Wilhelm, and U. Barth. Energy Efficiency of LTE networks under traffic loads of 2020. In *Proceedings of the Tenth International Symposium on Wireless Communication Systems (ISWCS)*, 2013.

[BDAK14] F. Beister, M. Dräxler, J. Aelken, and H. Karl. Power model design for ICT systems – A generic approach. *Computer Communications*, 50:77–85, September 2014.

[Bei14] F. Beister. Analyzing user video download behavior with clustering and hidden markov models. In *1st KuVS Workshop on Anticipatory Networks*, 2014.

[BFW+13] J. Bartelt, G. Fettweis, D. Wubben, M. Boldi, and B. Melis. Heterogeneous backhaul for cloud-based mobile networks. In *IEEE 78th Vehicular Technology Conference (VTC Fall)*, 2013.

[BHK13] A. Bokani, M. Hassan, and S. Kanhere. HTTP-Based Adaptive Streaming for Mobile Clients using Markov Decision Process. In *Packet Video Workshop (PV), 2013 20th International*, 2013.

[Bie12] T. Biermann. *Dealing with Backhaul Network Limitations in Coordinated Multi-point Deployments*. PhD thesis, University of Paderborn, Germany, 2012.

[BK14] F. Beister and H. Karl. Predicting mobile video inter-download times with hidden markov models. In *7th IEEE International Workshop on Selected Topics in Wireless and Mobile computing (STWiMob 2014)*, 2014.

[BLM+14] N. Bhushan, Junyi Li, D. Malladi, R. Gilmore, D. Brenner, A. Damnjanovic, R. Sukhavasi, C. Patel, and S. Geirhofer. Network densification: the dominant theme for wireless evolution into 5G. *Communications Magazine, IEEE*, 52(2):82–89, February 2014.

[BMKL12] M. Bansal, J. Mehlman, S. Katti, and P. Levis. OpenRadio: a programmable wireless dataplane. In *Proc. of the first workshop on Hot topics in software defined networks*, 2012.

[BMW14] N. Bui, F. Michelinakis, and J. Widmer. A Model for Throughput Prediction for Mobile Users. In *Proceedings of 20th European Wireless Conference*, 2014.

[BPC+05] A. Banerjee, Y. Park, F. Clarke, H. Song, S. Yang, G. Kramer, K. Kim, and B. Mukherjee. Wavelength-division-multiplexed passive optical network (WDM-PON) technologies for broadband access: a review [Invited]. *Journal of optical networking*, 4(11):737–758, 2005.

[BSC+11] T. Biermann, L. Scalia, C. Choi, H. Karl, and W. Kellerer. Backhaul Network Pre-Clustering in Cooperative Cellular Mobile Access Networks. In *Proc. IEEE World of Wireless Mobile and Multimedia Networks (WoWMoM)*, 2011.

[BSC+12] T. Biermann, L. Scalia, C. Choi, H. Karl, and W. Kellerer. CoMP clustering and backhaul limitations in cooperative cellular mobile access networks. *Pervasive and Mobile Computing*, 8(5):662–681, 2012.

[BSWK11] T. Biermann, L. Scalia, J. Widmer, and H. Karl. Backhaul Design and Controller Placement for Cooperative Mobile Access Networks. In *Proc. IEEE Vehicular Technology Conference (VTC-Spring)*, 2011.

[BV14] N. Barmann and S. Valentin. Wireless link quality prediction using street and coverage maps. In *1st KuVS Workshop on Anticipatory Networks*, 2014.

[BV15] W. Bao and S. Valentin. Bitrate adaptation for mobile video streaming based on buffer and channel state. In *Proceedings of the IEEE International Conference on Communications (ICC)*, 2015.

[BW14] N. Bui and J. Widmer. Modelling Throughput Prediction Errors as Gaussian Random Walks. In *1st KuVS Workshop on Anticipatory Networks*, 2014.

[Cis12] Cisco Visual Networking Index: Forecast and Methodology, 2011-2016. `http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360.pdf`, 2012.

[Cis14a] Cisco Visual Networking Index: Forecast and Methodology, 2013–2018. `http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.pdf`, 2014.

[Cis14b] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2013–2018. `http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.pdf`, 2014.

[CK00] C. Chekuri and S. Khanna. A PTAS for the multiple knapsack problem. In *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, 2000.

[CMD+13] C. Cicconetti, A. Morelli, M. Dräxler, H. Karl, V. Mancuso, V. Sciancalepore, R. Gupta, A. de la Oliva, I. Sánchez, P. Serrano, and L. Roullet. The Playground of Wireless Dense Networks of the Future. In *Future Network and MobileSummit 2013*, 2013.

[DBD+14] M. Dräxler, J. Blobel, P. Dreimann, S. Valentin, and H. Karl. Anticipatory Buffer Control and Quality Selection for Wireless Video Streaming. *arXiv preprint arXiv:1309.5491v2*, 2014.

[DBD+15] M. Dräxler, J. Blobel, P. Dreimann, S. Valentin, and H. Karl. SmarterPhones: Anticipatory Download Scheduling for Wireless Video Streaming. In *Proceedings of the International Conference on Networked Systems (NetSys)*, 2015.

[DBK+12] M. Dräxler, F. Beister, S. Kruska, J. Aelken, and H. Karl. Using OMNeT++ for Energy Optimization Simulations in Mobile Core Networks. In *Proceedings of the 5th International ICST Conference on Simulation Tools and Techniques (SIMUTOOLS)*, 2012.

[DBK13] M. Dräxler, T. Biermann, and H. Karl. Improving Cooperative Transmission Feasibility by Network Reconfiguration in Limited Backhaul Networks. *International Journal of Wireless Information Networks*, 20(3):183–194, 2013.

[DBK15] M. Dräxler, J. Blobel, and H. Karl. Anticipatory Download Scheduling in Wireless Video Streaming with Uncertain Data Rate Prediction. In *Proceedings of the 8th IFIP Wireless and Mobile Networking Conference (WMNC)*, 2015. submitted.

[DBKK12] M. Dräxler, T. Biermann, H. Karl, and W. Kellerer. Cooperating Base Station Set Selection and Network Reconfiguration in Limited Backhaul Networks. In *Proceedings of the IEEE 23nd International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2012.

[DDK14] M. Dräxler, P. Dreimann, and H. Karl. Anticipatory Power Cycling of Mobile Network Equipment for High Demand Multimedia Traffic. In *IEEE Online Conference on Green Communications (IEEE Online GreenComm'14)*, 2014.

[DK13] M. Dräxler and H. Karl. Cross-Layer Scheduling for Multi-Quality Video Streaming in Cellular Wireless Networks. In *Proceedings of the 9th IEEE International Wireless Communications and Mobile Computing Conference (IWCMC)*, 2013.

[DK14a] M. Dräxler and H. Karl. Feasibility of Base Station Coordination and Dynamic Backhaul Network Configuration in Backhaul Networks with Limited Capacity. In *European Wireless 2014 (EW2014)*, 2014.

[DK14b] M. Dräxler and H. Karl. SmarterPhones: Anticipatory Download Scheduling for Segmented Wireless Video Streaming. In *1st KuVS Workshop on Anticipatory Networks*, 2014.

[DK15] M. Dräxler and H. Karl. Dynamic Backhaul Network Configuration in SDN-based Cloud RANs. *arXiv preprint arXiv:1503.03309*, 2015.

[dlOMM+12] A. de la Oliva, A. Morelli, V. Mancuso, M. Dräxler, T. Hentschel, T. Melia, P. Seite, and C. Cicconetti. Denser networks for the Future Internet, the CROWD approach. In *MONAMI OConS Workshop: Workshop on Open Connectivity Services for the Future Internet*, 2012.

[DMMS14] S. Deb, P. Monogioudis, J. Miernik, and J. P. Seymour. Algorithms for enhanced inter-cell interference coordination (eicic) in lte hetnets. *IEEE/ACM Transations on Networking*, 22(1):137–150, February 2014.

[Dom00] G. Dommety. Key and Sequence Number Extensions to GRE. RFC 2890 (Proposed Standard), September 2000.

[DSK13] A. De Domenico, V. Savin, and D. Ktenas. A backhaul-aware cell selection algorithm for heterogeneous cellular networks. In *IEEE 24th International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC)*, 2013.

[FMHG08] R. Forsati, M. Mahdavi, A.T. Haghighat, and A. Ghariniyat. An efficient algorithm for bandwidth-delay constrained least cost multicast routing. In *Proceedings of the Canadian Conference on Electrical and Computer Engineering (CCECE)*, May 2008.

[GE08] K. Grobe and J. P. Elbers. PON in adolescence: from TDMA to WDM-PON. *Communications Magazine, IEEE*, 46(1):26–34, 2008.

[Goo] Google Inc. Android 3.0 Highlights. `http://developer.android.com/about/versions/android-3.0-highlights.html`.

[GRA+11] K. Grobe, M. Roppelt, A. Autenrieth, J.-P. Elbers, and M. H. Eiselt. Cost and energy consumption analysis of advanced WDM-PONs. *IEEE Communications Magazine*, 49(2), 2011.

[Gre13a] GreenTouch. GreenTouch Green Meter Research Study: Reducing the Net Energy Consumption in Communications Networks by up to 90% by 2020, 2013.

[Gre13b] GreenTouch. Mobile Communications WG Architecture Doc2: Reference scenarios, Version 1.3, 2013.

[Gre14] GreenTouch. Mobile Communications WG Architecture Doc2a: Update on Modelling Parameter, Version 1.3, 2014.

[GUR+13] S. Göndör, A. Uzun, T. Rohrmann, J. Tan, and R. Henniges. Predicting User Mobility in Mobile Radio Networks to Proactively Anticipate Traffic Hotspots. In *MOBILWARE'13*, 2013.

[HJM13] T.-Y. Huang, R. Johari, and N. McKeown. Downton Abbey Without the Hiccups: Buffer-based Rate Adaptation for HTTP Video Streaming. In *Proceedings of the 2013 ACM SIGCOMM Workshop on Future Human-centric Multimedia Networking*, 2013.

[HK98] C.-H. Hsu and U. Kremer. Iperf: A framework for automatic construction of performance prediction models. In *Proceedings of the Workshop on Profile and Feedback-Directed Compilation (PFDC)*, 1998.

[HSH+11] T. Hossfeld, M. Seufert, M. Hirth, T. Zinner, P. Tran-Gia, and R. Schatz. Quantification of YouTube QoE via Crowdsourcing. In *Proceedings of the IEEE International Symposium on Multimedia (ISM)*, 2011.

[HSS+15] Tobias Hoßfeld, Michael Seufert, Christian Sieber, Thomas Zinner, and Phuoc Tran-Gia. Identifying qoe optimal adaptation of {HTTP} adaptive streaming based on subjective studies. *Computer Networks*, (0):–, 2015. In Press, Uncorrected Proof.

[HSSZ14] T. Hossfeld, M. Seufert, C. Sieber, and T. Zinner. essing effect sizes of influence factors towards a qoe model for http adaptive streaming. In *Proceedings of the 6th International Workshop on Quality of Multimedia Experience (QoMEX)*, 2014.

[HZM14] J. Hao, R. Zimmermann, and H. Ma. GTube: Geo-predictive Video Streaming over HTTP in Mobile Environments. In *Proceedings of the 5th ACM Multimedia Systems Conference*, 2014.

[JMJ+13] V. Jungnickel, K. Manolakis, S. Jaeckel, M. Lossow, P. Farkas, M. Schlosser, and V. Braun. Backhaul requirements for inter-site cooperation in heterogeneous LTE-Advanced networks. In *IEEE International Conference on Communications Workshops (ICC)*, 2013.

[KPP04] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, 2004.

[KPS+06] S. Khan, Y. Peng, E. Steinbach, M. Sgroi, and W. Kellerer. Application-driven cross-layer optimization for video streaming over wireless networks. *IEEE Comm. Magazine*, 44(1):122–130, 2006.

[Law79] E. L. Lawler. Fast Approximation Algorithms for Knapsack Problems. *Mathematics of Operations Research*, 4(4):pp. 339–356, 1979.

[Law13] W. Law. Delivering Over The Top Video at Scale - Akamai at OTTCon 2013, 2013.

[LdV13] Z. Lu and G. de Veciana. Optimizing Stored Video Delivery For Mobile Networks: The Value of Knowing the Future. In *Proc. of the IEEE Int. Conf. on Comp. Comm. (INFOCOM)*, 2013.

[LHM10] B. Lantz, B. Heller, and N. McKeown. A network in a laptop: rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, 2010.

[LSJ+13] C. Liu, K. Sundaresan, M. Jiang, S. Rangarajan, and G.-K. Chang. The case for re-configurable backhaul in cloud-ran based small cell networks. In *Proceedings of the 2013 IEEE INFOCOM*, 2013.

[LZG+14] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A.C. Begen, and D. Oran. Probe and Adapt: Rate Adaptation for HTTP Video Streaming At Scale. *Selected Areas in Communications, IEEE Journal on*, 32(4):719–733, April 2014.

[MAB+08] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, March 2008.

[MATW15] K. Miller, A.-K. Al-Tamimi, and A. Wolisz. Low-Delay Adaptive Video Streaming Based on Short-Term TCP Throughput Prediction. *arXiv preprint arXiv:1503.02955v2*, 2015.

[MF07] P. Marsch and G. Fettweis. A Framework for Optimizing the Uplink Performance of Distributed Antenna Systems under a Constrained Backhaul. In *Proc. IEEE European Wireless Conference (EW)*, 2007.

[Mit02] T. M. Mitchell. *Machine learning*. McGraw-Hill, New York, 2002.

[MLT12] C. Müller, S. Lederer, and C. Timmerer. An evaluation of dynamic adaptive streaming over HTTP in vehicular environments. In *Proc. of the 4th Workshop on Mobile Video*, 2012.

[MQGW12] K. Miller, E. Quacchio, G. Gennari, and A. Wolisz. Adaptation algorithm for adaptive streaming over http. In *Packet Video Workshop (PV), 2012 19th International*, May 2012.

[MTVG14] J. Medved, A. Tkacik, R. Varga, and K. Gray. Opendaylight: Towards a model-driven sdn controller architecture. In *A World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014 IEEE 15th International Symposium on*, 2014.

[OAJ14] J. Otwani, A. Agarwal, and A. Jagannatham. Optimal scalable video scheduling policies for real time single and multiuser wireless video networks. *IEEE Transactions on Vehicular Technology*, PP(99):1–1, 2014.

[ODL] OpenDayLight: Technical Overvew. `http://www.opendaylight.org/project/technical-overview`. Accessed: 26.04.2015.

[OF 13] OpenFlow Switch Specification, Version 1.4.0 (Wire Protocol 0x05). `https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf`, 2013.

[OFGZ13] J.J. Olmos, R. Ferrus, and H. Galeana-Zapien. Analytical Modeling and Performance Evaluation of Cell Selection Algorithms for Mobile Networks with Backhaul Capacity Constraints. *Wireless Communications, IEEE Transactions on*, 12(12):6011–6023, 2013.

[ONF12] Software-Defined Networking: The New Norm for Networks. `https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf`, 2012.

[ONF13] SDN Architecture Overview. `https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/SDN-architecture-overview-1.0.pdf`, 2013.

[PGH08] A. Papadogiannis, D. Gesbert, and E. Hardouin. A Dynamic Clustering Approach in Wireless Networks with Multi-Cell Cooperative Processing. In *Proc. IEEE International Conference on Communications (ICC)*, 2008.

[PMA13] R. Pantos, W. May, and Apple Inc. HTTP Live Streaming. `http://tools.ietf.org/html/draft-pantos-http-live-streaming-11`, April 2013.

[RBV+12] H. Riiser, H. S. Bergsaker, P. Vigmostad, P. Halvorsen, and C. Griwodz. A comparison of quality scheduling in commercial adaptive HTTP streaming solutions on a 3G network. In *Proc. of the 4th Workshop on Mobile Video*, 2012.

[REV+12] H. Riiser, T. Endestad, P. Vigmostad, C. Griwodz, and P. Halvorsen. Video streaming using a location-based bandwidth-lookup service for bitrate planning. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 8(3):1–19, July 2012.

[RN12] R. Radhakrishnan and A. Nayak. Cross layer design for efficient video streaming over LTE using scalable video coding. In *Communications (ICC), 2012 IEEE International Conference on*, 2012.

[RSS10] R. Ramaswami, K.N. Sivarajan, and G.G.H. Sasaki. *Optical Networks: A Practical Perspective*. Morgan Kaufmann. Elsevier/Morgan Kaufmann, 2010.

[RTN12] R. Radhakrishnan, B. Tirouvengadam, and A. Nayak. Channel quality-based AMC and smart scheduling scheme for SVC video transmission in LTE MBSFN networks. In *Communications (ICC), 2012 IEEE International Conference on*, 2012.

[RVGH13] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen. Commute path bandwidth traces from 3G networks: analysis and applications. In *Proc. of the 4th ACM Multimedia Sys. Conf.*, 2013.

[SAD+15] M. I. Sanchez, A. Asadi, M. Dräxler, R. Gupta, V. Mancuso, A. Morelli, A. de la Oliva, and V. Sciancalepore. Tackling the increased density of 5G networks; the CROWD approach. In *IEEE 81st Vehicular Technology Conference: VTC2015-Spring, First International Workshop on 5G Architecture (5GArch 2015)*, 2015.

[SES+14] 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*, PP(99), 2014.

[SKM+10] M. Sawahashi, Y. Kishiyama, A. Morimoto, D. Nishikawa, and M. Tanno. Coordinated multipoint transmission/reception techniques for LTE-advanced [Coordinated and Distributed MIMO]. *IEEE Wireless Communications*, 17(3):26–34, 2010.

[SL02] A. Sang and S. Li. A predictability analysis of network traffic. In *Computer networks*, 2002.

[SMW07] H. Schwarz, D. Marpe, and T. Wiegand. Overview of the scalable video coding extension of the h.264/avc standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9):1103–1120, Sept 2007.

[SNK13] H. M. Soliman, O. A. Nasr, and M. M. Khairy. Analysis and optimization of backhaul sharing in CoMP. In *IEEE 24th International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC)*, 2013.

[SPJFL15] B. Soret, K.I. Pedersen, N.T.K. Jorgensen, and V. Fernandez-Lopez. Interference coordination for dense wireless networks. *IEEE Communications Magazine*, 53(1):102–109, January 2015.

[SQBB10] C. Song, Z. Qu, N. Blumm, and A.-L. Barabási. Limits of predictability in human mobility. *Science*, 327(5968):1018–1021, 2010.

[SRV02] H.F. Salama, D.S. Reeves, and Y. Viniotis. Evaluation of multicast routing algorithms for real-time communication on high-speed networks. *Selected Areas in Communications*, 15(3):332–345, 2002.

[SSPS09] O. Simeone, O. Somekh, H. V. Poor, and S. Shamai. Downlink multicell processing with limited-backhaul capacity. *EURASIP Journal on Advances in Signal Processing*, 2009.

[TMF+14] S. Tombaz, P. Monti, F. Farias, M. Fiorani, L. Wosinska, and J. Zander. Is backhaul becoming a bottleneck for green wireless access networks? In *Proceedings of the IEEE International Conference on Communications (ICC)*, June 2014.

[Twi] Twisted Matrix Labs. Twisted. `http://twistedmatrix.com/`.

[Val14] S. Valentin. Anticipatory resource allocation for wireless video streaming. In *Proceedings of the IEEE International Conference on Communication Systems (ICCS)*, 2014. invited paper.

[Vid] VideoLAN Organization. VideoLAN. `http://www.videolan.org/videolan/`.

[WDS+14] P. Wette, M. Dräxler, A. Schwabe, F. Wallaschek, M. Hassan Zahraee, and H. Karl. MaxiNet: Distributed Emulation of Software-Defined Networks. In *Proceedings of the 2014 IFIP Networking Conference (Networking 2014)*, 2014.

[WLA12] IEEE Standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007), 2012.

[WLLM13] R. Wang, H.H. Lee, S.S. Lee, and B. Mukherjee. Energy saving via dynamic wavelength sharing in WDM-PON. *ONDM*, pages 235–239, 2013.

[YKH08] J. Yao, S. S. Kanhere, and M. Hassan. An empirical study of bandwidth predictability in mobile computing. In *Proc. of the 3rd ACM Int. Workshop on Wireless network testbeds, experimental evaluation and characterization - WiNTECH*, 2008.

[YKH11] J. Yao, S. S. Kanhere, and M. Hassan. Mobile Broadband Performance Measured from High-Speed Regional Trains. In *Proc. of the IEEE Vehicular Technology Conference (VTC Fall)*, 2011.

[YKH12] J. Yao, S. S. Kanhere, and M. Hassan. Improving QoS in High-Speed Mobility Using Bandwidth Maps. *IEEE Trans. Mob. Comput.*, 11(4):603–617, 2012.

[ZBE01] W. Zhengying, S. Bingxin, and Z. Erdun. Bandwidth-delay-constrained least-cost multicast routing based on heuristic genetic algorithm. *Computer Communications*, 24(7-8):685–692, 2001.

[ZL12] J. Zhao and Z. Lei. Clustering methods for base station cooperation. In *Wireless Communications and Networking Conference (WCNC), 2012 IEEE*, 2012.

[ZYM12] Q. Zhang, C. Yang, and A.F. Molisch. Cooperative downlink transmission mode selection under limited-capacity backhaul. In *Wireless Communications and Networking Conference (WCNC), 2012 IEEE*, pages 1082–1087, 2012.