Quality of Experience-Aware Progressive Video Caching and Adaptive Device-to-Device Video Streaming for Cellular Networks with High User Density

by

Ala'a Al-Habashna

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Electrical and Computer Engineering

Carleton University
Ottawa, Ontario

Abstract

The increasing popularity of video streaming is escalating the growth of data traffic over cellular networks. Consequently, new techniques are much needed to help serving this increasing video traffic. Furthermore, the new techniques should consider the complex, dynamic, and delay-sensitive nature of video streaming traffic to support good Quality of Experience (QoE) video streaming services over cellular networks.

The work in this thesis is focused on proposing algorithms and techniques to enhance the delivery of video contents and to improve the QoE of video streaming over cellular networks with high user density. First, we propose two algorithms for progressive caching of video segments in User Equipments (UEs) and Device-to-Device (D2D) transmission of video contents among UEs in the cell. The algorithms are employed, by the Base-Station (BS), to send segments of video files to selected UEs in the cellular network (called Storage Members (SMs)), to cache and forward the segments to requesting UEs using D2D communication. We study the performance of both algorithms in terms of the hit ratio as well as the achieved data rates.

The parameters of the wireless communication on the Radio Access Network (RAN) between the BS and UEs have an effect on video streaming QoE. As such, we analyze the impact of the wireless transmission parameters in Long Term Evolution-Advanced (LTE-A) networks on video streaming QoE. We consider both cell-level and link-level parameters. Moreover, we propose an architecture for improving the QoE of video streaming in cellular networks with high user density. The architecture employs the aforementioned algorithms. Furthermore, the architecture employs Dynamic Adaptive

Streaming over HTTP (DASH); an adaptive video streaming technique. We study the improvements achieved by the proposed architecture in terms of many video streaming QoE metrics. Thereafter, we improve the operation of the proposed architecture by introducing QoE awareness to both caching and distribution of video segments. We employ QoE awareness in three aspects of the proposed architecture; cellular resource allocation, caching of video segments, and SM assignment optimization. We analyze the improvements achieved by each QoE awareness technique in terms of video streaming QoE metrics.

Acknowledgments

I would like to express my gratitude and thanks to my supervisor Professor Gabriel Wainer for his encouragement, mentorship, and technical guidance during the PhD program. His support and advice have helped me tremendously.

I would also like to thank Professor Stenio Fernandes from the Federal University of Pernambuco, Recife, Brazil, for his collaboration and insightful feedback which incented me to improve my research. Special thanks also go to Dr. Gary Boudreau and Ronald Casselman from Ericsson Canada for their assistance with parts of this this work.

Finally, and most importantly, I would like to express my deepest gratitude and appreciation to my parents: Saqer and Amneh, and to my siblings: Baha, Haneen, Sahar, and Nisreen. Throughout eleven years away from you that felt like eternity, and throughout the ups and downs from which I have emerged successful, your unconditional love and belief in me have always been a constant source of encouragement and inspiration. Thank you for your support, and this thesis is dedicated to you.

Table of Contents

List of Tables

List of Figures

List of Acronyms

AWGN: Additive White Gaussian Noise

BB: Buffer-Based scheduling

BB-HRC: BB with High Rate Caching

BB-HRC-SMA: BB-HRC with SM Assignment optimization

BS: Base-Station

CDN: Content Distribution Network

CQI: Channel Quality Indicator

CSVD: Cached and Segmented Video Download

DABAST: **DA**SH-based **B**S-**A**ssisted D2D video **ST**reaming

DASH: Dynamic Adaptive Streaming over HTTP

DEVS: Discrete EVent System Specification

DISCS: DIStributed Cached and Segmented video download

DL: DownLink

DTSMs: Distribute To SMs

D2D: Device-to-Device

ECDF: Empirical Cumulative Distribution Function

ICN: Information-Centric Networking

iCSVD: improved Cached and Segmented Video Download

LP: Linear Programming

LTE-A: Long Term Evolution-Advanced

MAC: Medium Access Control

MILP: Mixed Integer Linear Programming

MIQCP: Mixed Integer Quadratically Constrained Programming

MIQP: Mixed Integer Quadratic Programming

MoE: Margin of Error

MOS: Mean Opinion Score

NNs: Neural Networks

NSM: Non-Storage Member

OFDMA: Orthogonal Frequency-Division Multiple Access

OTT: Over The Top

PDCP: Packet Data Convergence Protocol

PF: Proportional Fair

P2P: Peer-to-Peer

QCP: Quadratically Constrained Programming

QoE: Quality of Experience

QoS: Quality of Service

QP: Quadratic Programming

RAN: Radio Access Network

RATs: Radio Access Technologies

RB: Resource Block

RLC: Radio Link Control

RR: Round Robin

SM: Storage Member

STSM: Send To an SM

STUE: Send To a UE

SVD: Segmented Video Download

SWA: Send With Assistance

TTI: Transmission Time Interval

UE: User Equipment

UL: UpLink

UML: Unified Modeling Language

V&V: Verification and Validation

5G: Fifth Generation

Chapter 1

Introduction

Cellular networks have witnessed a continuous growth in data traffic and increasing demand for higher data rates due to the improvements on mobile devices, the services provided, and the increasing number of users [1]. Satisfying these demands has become a challenge for cellular network operators. The scarcity of the radio spectrum in cellular networks makes it difficult to provide all the users with enough resources to achieve the desired performance. Despite the emergence of new techniques to increase spectrum utilization such as opportunistic spectrum sharing and cognitive radio [2], [3], the radio spectrum is still considered scarce. Innovative communication techniques and algorithms are still needed to improve performance.

This situation is made worse by the increasing popularity of video applications and the improvements on the screen size and quality of smart devices. Wireless and mobile users increasingly use their phones and tablets to watch videos, which increased video traffic over cellular networks. Video traffic accounted for 60% of the total mobile data traffic in 2016 [4], and it is expected to account for over three-fourths of the world's mobile data traffic by 2021 [4]. Hence, new techniques are needed to help serving video traffic over cellular networks, which is becoming the main source of data traffic.

Device-to-Device (D2D) communication, introduced by the Long Term Evolution-Advanced (LTE-A) standard [5], is a new communication paradigm that allows direct

communication between nearby User Equipments (UEs) without routing the traffic through the Base-Station (BS) and the network infrastructure. This direct communication between UEs provides many benefits. The capacity of the network can be improved due to the reuse of frequency resources, as D2D transmission usually takes place over short distance. Moreover, this short distance transmission potentially has favorable propagation conditions, which leads to data rate gains. D2D communication can also be utilized to extend the cell coverage and improve the service for users at the cell-edge. As such, D2D communication is an important technology that is adopted by the Fifth Generation (5G) cellular networks [6], and much research has been conducted in recent years on D2D communication and developing applications for it in cellular networks [7]–[11].

We propose two algorithms for BS-assisted progressive caching of video segments in UEs and Peer-to-Peer (P2P) transmission of video contents among UEs using D2D communication, to improve video transmission in cellular networks [12]–[16]. Our algorithms are inspired by the architecture proposed in [17], which exploits D2D communications for video transmission improvement. The main idea is to utilize popular video contents that is cached in the UE devices. If cached video files are requested, they will be sent to the requesting devices from the caching UEs over D2D links.

The proposed algorithms do not only consider the transmission of video segments to requesting UEs over D2D links, but also caching of video segments. Locally-popular videos can be strategically cached to be available to all users in the cell. Furthermore, the proposed algorithms progressively cache video segments as requested, and do not assume that all contents can be cached well in advance. Although these characteristics are very beneficial in any video streaming scenario with high number of users, they are crucial for

high traffic load scenarios where users are requesting recently published contents. Such contents are usually transmitted to requesting users over cellular resources because they are recently published (not cached). As such, under high traffic load, it would be highly inefficient to transmit unrequested video contents for caching under the assumption they might be requested in the future, as this would waist precious cellular resources. At the same time, requested video segments (especially popular ones) should be cached in case they are requested later. Such scenario is common nowadays because many video streaming websites, including YouTube, provide a service that is a combination of live and on-demand video streaming. YouTube live is an example on such service. With YouTube live, live videos are automatically recorded and saved on the user's YouTube channel. This means that during the live video stream, viewers can rewind the live stream to watch any parts they might have missed or want to watch again. Once the live stream is complete, the whole video stream will be available at the user's YouTube channel. Many popular videos nowadays are published using this service. This includes sport press conferences, political events such as debates, and popular podcasts. This provides very popular use cases for the progressive caching in our cached and segmented video download algorithms. As popular content is published live, this provides the first motivation, which is a video stream with a large number of viewers that are likely within proximity of each other, such as fans around an arena and watching a post sport-event press conference. Second, such content can not be cached in advance as it is published live. Moreover, because viewers do not have to be synchronized as in the traditional live streaming case, many viewers might start watching the beginning of the stream, shortly after the actual start time of the stream. Even if the stream is over, it will be available on the

YouTube channel, so other users might join towards the end. As such, viewers who started watching around the beginning of the event can cache video segments. These video segments will be utilized by viewers who start watching later, after the beginning of the event.

The first algorithm is called *Cached and Segmented Video Download* (CSVD) [12], [14]–[16]. In CSVD, the cell is divided into clusters. Selected UEs in each cluster are chosen as Storage Members (SMs). When an SM requests a video file, the BS sends segments of the video file to the SM over a cellular link. The SM will be requested to cache the downloaded file. When a cached file is requested by other UEs in the cluster, the segments will be sent over D2D links from the caching SM.

The second algorithm is called *DIStributed, Cached, and Segmented video download* (DISCS) [12], [14]–[16]. In DISCS, a popular video file is also divided into segments, which are distributed over multiple SMs to be cached and forwarded to the requesting UE. This provides further parallelism when transmitting video files and further load balancing among SMs, which speeds up the transmission process. Furthermore, in DISCS, the SMs might be required to receive and cache segments they have not requested when asked for assistance (as opposed to just forwarding pieces they already have), which helps accumulating video files faster in the distributed cache.

The algorithms define how video segments are cached and exchanged among the UEs. A protocol has been developed, including a variety of messages necessary for this communication, and a complete definition of the protocol is described. With the proposed algorithms, SMs are assigned to requesting UEs on a segment-by-segment basis (rather than permeant pairing) depending on many factors, such as the availability of segments

and SMs. A new feature of the algorithms is that they employ new technologies in the 5G cellular networks such as multi-Radio Access Technologies (RATs), dual connectivity, and flow splitting/aggregation [18], [19] to send multiple video segments to a user (from more than one source) simultaneously, when possible.

We developed analytical models to evaluate the efficiency of caching video files in UEs with both algorithms in terms of the hit ratio. Analytical closed-from expressions were derived for the hit ratio of both algorithms. Furthermore, we built a suite of models using the Discrete EVent System Specification (DEVS) formalism [20], [21] to model LTE-A cellular networks that employ CSVD and DISCS. System-level simulations were performed to evaluate the performance of CSVD and DISCS with different parameters and under different simulation scenarios. Simulation results show that the proposed algorithms can achieve significant improvements over conventional transmission methods in terms of both the cell's aggregate data rate as well as the average data rate per user. After developing the algorithms above to improve video contents transmission over cellular networks, we focus on video streaming over cellular networks, as most videos on the web nowadays are accessed via streaming.

Video streaming provides a convenient way to watch online video contents. With video streaming, a user can start playing the video before the entire video file is downloaded. Over the last few years, many platforms for live and on-demand video streaming services have emerged and become very popular such as YouTube, Netflix, and Hulu. With such platforms, users can watch their favourite shows at anytime and anywhere. Furthermore, some of these services are free (e.g., YouTube) and others cost less than 10 dollars a month. As such, many people nowadays are cancelling their cable TV contracts to

subscribe for such video streaming services instead. Moreover, TV networks like ESPN provide live streaming of sport events for users to watch online. Video streaming has revolutionized education. For example, YouTube is full of tutorials and lectures that are available for people around the world. All this has caused video streaming to gain in popularity to the point where it has changed the way we consume entertainment and information. As per [22], IP video traffic is expected to account for 82 percent of the overall Internet traffic by 2021. At the same time, the improvement on the performance as well as the screen size and quality of mobile devices, such as mobile phones and tablets, has caused users to increasingly watch online videos on their smartphones and tablets. This has caused video streaming to be the most popular application on cellular networks. According to [23], during the peak hours, YouTube traffic only accounts for 21% of the mobile downlink video traffic in North America.

Supporting good Quality of Experience (QoE) video streaming services and satisfying these increasing demands are challenging for cellular network operators due to many reasons. The scarcity of the radio spectrum in cellular networks makes it difficult to provide the necessary data rates for users to enjoy high QoE video streaming. Furthermore, the transmission impairments in cellular wireless communication such as path-loss and fading further decrease the data rates of cellular links. This could cause frequent video stalling, which significantly degrades the end-user QoE. As such, the parameters of the cellular transmission over which video segments are delivered to UEs have an impact on the end-user QoE.

The challenge of video streaming over networks with limited capacity and variable network conditions motivated the idea of Dynamic Adaptive Streaming over HTTP

(DASH) [24]–[27]. DASH is an adaptive video streaming technique that provides efficient bandwidth utilization and improved streaming quality. In DASH, a video is divided into small segments, and each segment is encoded in multiple video bit rates. This provides different quality levels and allows adaptive streaming. Due to its advantages, DASH has been employed by big video streaming platforms, such as YouTube and Netflix, and it is being adopted by an increasing number of video applications. As such, we consider DASH-based video streaming in our work.

Although DASH provides an improvement for video streaming over networks with highly-variable data rates, the limited capacity of cellular networks is still a problem when it comes to providing good QoE video streaming over cellular networks. As such, we first study how much the cellular transmission parameters affect the QoE of video streaming. Then we propose a solution to the problem.

We study the impact of transmission parameters in cellular networks on multiple video streaming QoE metrics [28]. We consider parameters at both the cell level (cell bandwidth and number of UEs in the cell) as well as the link level (BS transmission power and BS-UE distance). We built a DEVS model for video streaming over an LTE-A network, and used the model to run various simulations under different scenarios. We provide an exploratory data analysis of the results to evaluate the effect of the transmission parameters on many QoE metrics, such as video stalling and initial delay.

In the next chapter, we present some related work that considers the effect of the wireless link conditions on video streaming QoE. However, to the best of our knowledge, this is the first work that considers the impact of transmission parameters at both the cell level

and link level on video streaming QoE. This is important because operators are changing their perspective from network-centric to experience-centric operation. As video streaming is very popular over cellular networks, providing good QoE video service is a key to differentiated competitiveness. Hence, this study provides a valuable insight for LTE-A network design, especially in terms of small cells deployment and configuration in urban areas (where there is usually high user density [29]).

Results for the impact of transmission parameters on video streaming QoE show that cell congestion and/or lack of resources could cause serous degradation to video streaming QoE. As such, we propose and evaluate an architecture that improves the QoE of video streaming over cellular networks in such scenarios [30]–[32]. The proposed architecture employs the following techniques:

- The CSVD and DISCS algorithms presented above, which provide BS-assisted progressive caching of video segments and D2D video transmission in cellular networks. Here, CSVD and DISCS are adopted in the context of video streaming. We evaluate the performance of the proposed architecture in terms of video streaming QoE metrics

- DASH is also employed to allow adaptive video streaming. This is employed in our architecture due to its advantages and to allow support for DASH-based applications

The proposed architecture is called DABAST: **DA**SH-based **B**S-**A**ssisted D2D video **ST**reaming in cellular networks. Such architecture improves the performance of video streaming over cellular networks with high user density. With DABAST, we consider the

case of progressive caching of video segments under high traffic load. As discussed above, this is a common scenario where many UEs could be requesting a recently published popular content under high traffic load. We also consider the case where there is an overloaded and limited cellular channel that provides access to any video segment at any available video bit rate, and an out-of-band D2D channel with much higher bandwidth that only provides access to certain videos (at certain video bit rates). DABAST provides a system that makes the best of available channels and cached segments. With DABAST, SMs are assigned to requesting UEs on a segment-by-segment basis. Many decisions need to be taken dynamically regarding caching and distribution of video contents, as well as allocation of resources and SMs to improve video streaming QoE for users in the cell. Another feature of DABAST is that it can operate in proactive mode, where up to a certain number of segments can be sent to the client proactively. We provide a detailed description for DABAST in Chapter 7.

We use the DEVS formalism to build a model for the proposed architecture in an LTE-A network. Simulations based on this model are used to evaluate the performance of DABAST in terms of video streaming QoE metrics. Simulation results show that the proposed architecture achieves significant improvements in terms of QoE when compared to conventional video streaming over a cellular network, i.e., DASH streaming without D2D/P2P streaming. We also study the impact of the caching and distribution algorithm (CSVD versus DISCS) on the performance improvements achieved by DABAST. We provide a thorough analysis of the results which brings very interesting findings about the impact of the employed algorithm on video streaming QoE in high user density scenarios.

As mentioned above, simulation results show that DABAST achieves significant improvements in terms of video streaming QoE metrics for users in the cell. However, the results also show that despite the significant improvements achieved by DABAST, some users do not receive video streaming service with good QoE, due to repetitive video rebuffering. Furthermore, some results show that significant increase in the QoS of users does not always translate into a significant increase in their QoE due to the different factors involved in video streaming QoE. As such, the operation of DABAST can be further improved by making it aware of video streaming QoE for users in the cell. This can further increase the QoE gains achieved by DABAST by targeting users who are experiencing poor QoE or by improving a certain QoE metric as needed. We employ QoE awareness in three aspects of DABAST, namely, cellular resource allocation, caching of video segments, and SM assignment optimization; where the optimizer dynamically and adaptively maximizes the video bit rate while minimizing the number of rebufferings in the cell. We analyze the improvement achieved by each QoE awareness technique in terms of video streaming QoE metrics. Results show that all the QoE awareness techniques above do indeed improve the performance gains achieved by DABAST.

1.1 Thesis objectives

Here, we provide a summary of the thesis objectives. The first goal of this thesis is to provide approaches for improving the transmission of video contents over cellular networks with high user density and in the presence of popular video contents. Because video traffic is becoming the majority of data traffic over cellular networks, improving its transmission and increasing its throughput will improve the overall throughput of the

cellular network. We aim to provide algorithms that consider not only video files distribution, but also progressive caching. Under high user density and high traffic load, such approaches progressively and strategically cache popular videos to be available for all users in the cell in a way that makes efficient use of cellular resources. This is crucial for scenarios with high traffic load where users are requesting recently published videos. We also aim to study the performance and improvements achieved by such algorithms.

As most of videos on the internet are accessed via streaming (YouTube, NetFlix, Hulu, etc.), we study video streaming over cellular networks. The Radio Access Network (RAN) is the main bottleneck in cellular networks with limited frequency resources that are shared among large number of users. As such, we first want to analyze the impact of transmission parameters over the RAN on the QoE of DASH-based video streaming.

The lack of enough resources in the RAN could cause a significant degradation to video streaming QoE under high traffic load. As such, another goal is to provide an architecture that utilizes the proposed algorithms above to improve DASH-based video streaming QoE over cellular networks. Another goal is to make this architecture aware of the QoE of users in the network and consider it in video caching and distribution to maximize video streaming QoE for users in the cell.

1.2 Thesis contributions

Here we summarize the main contributions of this thesis:

- An algorithm, namely CSVD, for progressive caching and D2D distribution of video segments to improve video transmission over cellular networks

- A second algorithm, namely DISCS, for progressive caching and D2D distribution of video segments to further improve video transmission over cellular networks
- Analytical closed-form expressions for the hit ratio of video caching with both algorithms
- A DEVS model for an LTE-A network that implements CSVD and DISCS
- An implementation of the LTE-A network DEVS model using the CD++ toolkit to study the performance of the proposed algorithms
- A thorough analysis for the impact of the transmission parameters at the cell level in addition to the link-level parameters on many video streaming QoE metrics
- An architecture, namely DABAST, which employs the aforementioned algorithms and DASH to improve the QoE of video streaming in cellular networks
- A DEVS model for DABAST in an LTE-A network, and an implementation of the DEVS model using the CD++ toolkit to study DABAST's performance
- Analysis for the impact of the caching and distribution algorithm on the performance improvement achieved by DABAST
- Employment of QoE-based cellular resource allocation in DABAST and analyzing the improvement achieved over state-of-the-art cellular resource allocation algorithms

- An approach for high video bit rate caching of video segments in DABAST to further improve video bit rate under relatively lower traffic load

- An optimization model for the SM assignment problem in DABAST to dynamically and adaptively maximize the aggregate video bit rate in the cell and minimize the number of rebufferings

- Integration of the CD++ toolkit with a commercial optimization solver to run the proposed optimization model in the simulations, and evaluate the improvement achieved by the proposed SM assignment optimization approach

As a result of this research, the following patent, papers, and articles have been filed, published, or submitted for publication:

- A. Al-Habashna, G. Wainer, G. Boudreau, and R. Casselman. "Improving wireless video transmission in cellular networks using D2D communication." Canada. Provisional patent filing reference P47111. May 2015.

- A. Al-Habashna, G. Wainer, G. Boudreau, and R. Casselman, "Cached and segmented video download for wireless video transmission," *in Proceedings of the 2016 Annual Simulation Symposium (ANSS'16)*, Pasadena, USA, 2016.

- A. Al-Habashna, G. Wainer, G. Boudreau, and R. Casselman, "Distributed cached and segmented video download for video transmission in cellular networks," *in the 2016 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS'16)*, Montreal, Canada, 2016.

- A. Al-Habashna, S. Fernandes, and G. Wainer, "DASH-based peer-to-peer video streaming in cellular networks," *in the 2016 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS'16)*, Montreal, Canada, 2016.
- A. Al-Habashna and G. Wainer, "Improving video transmission in cellular networks with cached and segmented video download algorithms," *Mobile Networks and Applications*, vol. 23, no. 3, Sept. 2017 (**impact factor 3.259**).
- A. Al-Habashna and G. Wainer, "DEVS-based modeling of cached and segmented video download algorithms in LTE-A cellular networks," *in Proceedings of the 20th Communications and Networking Symposium (CNS'17)*, Virginia Beach, USA, 2017.
- A. Al-Habashna, G. Wainer, and S. Fernandes, "Improving video streaming over cellular networks with DASH-based device-to-device streaming," *in the 2017 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS'17)*, Bellevue, USA, 2017.
- A. Al-Habashna, S. Fernandes, and G. Wainer, "Analyzing the effect of LTE-A transmission parameters on video streaming quality of experience," *in Proceedings of the 21st Communications and Networking Symposium (CNS'18)*, Baltimore, USA, 2018 (received the best paper award at CNS'18, and the runner-up for the overall best paper award at the SpringSim'18 multi-conference).
- A. Al-Habashna and G. Wainer, "DASH-based device-to-device video streaming for cellular networks with high user density," *in the 2018*

International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS'18), Bordeaux, France, 2017.

- A. Al-Habashna and G. Wainer, "QoE awareness in progressive video caching and DASH-based D2D video streaming in cellular networks," submitted to *Mobile Networks and Applications,* Jun. 2018.

In addition to the publications above which are all out of the work in this thesis, we worked on a crowd modeling project during my PhD program and published the following paper out of that project,

- A. Al-Habashna and G. Wainer, "Modeling pedestrian behavior with Cell-DEVS: theory and applications," *Simulation: Transactions of the Society for Modeling and Simulation International*, vol. 92, no. 2, Dec. 2015.

1.3 Thesis organization

The rest of this thesis is organized as follows, in Chapter 2, the technical background of the topics in this thesis are discussed and the related work in the literature is reviewed. In Chapter 3, the cached and segmented video download algorithms and their operation are discussed in detail. Furthermore, the developed analytical models for video caching with both algorithms are presented and closed-form expression for the hit ratio of both algorithms are derived. In Chapter 4, our cellular network DEVS model is presented. In Chapter 5, analytical and simulation results for the hit ratio of both algorithms are discussed. Thereafter, results for data rates of the proposed algorithms are analyzed. In Chapter 6, our video streaming DEVS model is presented and a thorough analysis for the impact of transmission parameters on video streaming QoE is provided. In Chapter 7,

DABAST and its operation are discussed. Performance evaluation of DABAST under various scenarios is provided after in that chapter. In Chapter 8, DABAST with QoE-aware cellular resource allocation is presented and simulation results for the performance gains achieved by that is discussed. In Chapter 9, DABAST with high rate caching is presented. Thereafter, SM assignment optimization in DABAST and the proposed optimization models are discussed. A review of the integration of CD++ and the Gurobi optimizer is provided. Afterwards, simulation results for the performance improvements achieved by high rate caching and SM assignment optimization are analyzed. Finally, in Chapter 10, conclusion and future work are listed.

Chapter 2

Technical background and related work

In this chapter, we discuss the technical background of the topics in this thesis. Afterwards we present the related work in the literature.

2.1 Technical background

2.1.1 Cellular networks and D2D communication

In recent years, the advance in cellular networks and mobile devices has led to major improvements in the services provided to cellular network users. This has caused the rate of adoption of mobile devices to grow rapidly. As per [1], the number of global mobile subscriptions has been increasing by more than 20% annually in the last 5 years, and it is expected to reach 4.3 billion subscriptions by the end of 2017. Due to this increasing number of users and to the newly provided bandwidth-demanding services, the demands for higher data rates has increased significantly. With this continuous growth of data traffic on cellular networks, the gap between capacity requirements and spectrum shortage is becoming a serious problem. This has made the cellular bandwidth bottleneck a key challenge for the 5G cellular networks. As the scarcity of the radio spectrum is the main reason for this bottleneck, spectral efficiency became a primary concern in the design of cellular data communication systems.

Cognitive radio was developed to provide a solution to the radio spectrum scarcity problem by exploiting the spectrum in more intelligent and flexible ways [2], [3].

Cognitive radio employs dynamic spectrum access techniques to allow users who have no spectrum licenses, also known as secondary users, to use the temporarily unused licensed spectrum. Another solution that has been implemented to increase the frequency efficiency and provide higher data rates is decreasing the size of cells [29], [33]. However, there are some challenges for having small-sized cells. First, it becomes more difficult to handle mobility and control interference as the size of the cells decreases. Second, it becomes more costly to implement a high-capacity wired backbone as the number of cells increases. In spite of the attempts above to increase the efficiency of spectral utilization, the growth in user demands are outpacing the networks capacity and new techniques are much needed to improve performance.

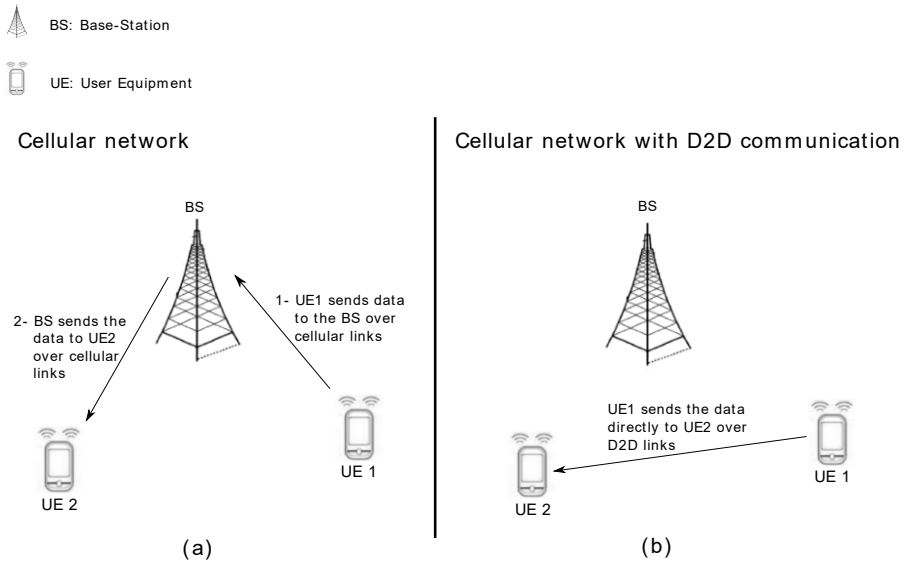

Figure 2.1. A cellular network (a) without D2D communication (b) with D2D communication.

D2D communication, introduced by the LTE-A standard [7]–[10], is a new communication paradigm that allows direct communication between nearby UEs without routing the traffic through the BS and the network infrastructure. In a traditional cellular network, all traffic from and to the UEs has to go through the BS, even if the routed

traffic is between two UEs who are within proximity of direct communication. Figure 2.1-a shows an example for such traditional approach. This way of communication is suitable for traditional mobile services that require low data rate such as voice calls and text messaging. Never the less, modern services over cellular networks such as content sharing and social networking provide many use cases for D2D communication when the communicating users are within range for direct communication, as can be seen in Figure 2.1-b. In such cases, D2D communications can highly increase the spectral efficiency of the network due to spatial reuse. Furthermore, D2D communication in such scenarios can potentially improve throughput and delay due to communication over one-hop and shorter distance. As such, D2D communication is an important technology that is adopted by the 5G cellular networks [6].

D2D communication can take place over the cellular spectrum (in-band) or over unlicensed spectrum (out-of-band) [8]. Some work in the literature on D2D communication proposes using in-band D2D communication. If the cellular spectrum is shared by both cellular and D2D communication, which is the case referred to as underlay in-band D2D, careful design of the system should be implemented to mitigate interference between D2D and cellular communication. Hence, many have proposed to separate the frequency resources dedicated for cellular communication from those dedicated for D2D communication (overlay in-band) to mitigate such interference. Out-of-band D2D communication has been proposed by many researches in order to further increase the network capacity and eliminate any interference between cellular and D2D communication. As such, we consider the case of out-of-band D2D communication in our research. D2D communications in cellular networks are expected to be controlled by the

BS [8]. The BS supervision is necessary in resolving challenges such as interference avoidance and synchronization.

D2D communication depends on the participation of users for sharing contents. As such, it is important to find approaches to incentivize and motivate the involvement of users in D2D communication. We do not consider incentive mechanisms here, as this is a different research area that is out of the scope of this thesis. There has been much research on incentive mechanisms to motivate such user involvement in D2D communication. For more information about the research in this area, the interested reader is referred to [34]–[36].

2.1.2 Video streaming

Video streaming is the most dominant application on the internet nowadays [23]. Recent traffic forecast studies [22] show that video accounted for 73 percent of the Internet traffic in 2016. Most videos on the web nowadays are accessed via streaming. Contents such as movies and TV shows, news video clips, YouTube videos, and sport events are all watched online by people around the world every day. Due to the continuously increasing popularity of video streaming applications and their high bandwidth requirements, video streaming has received much interest for over two decades [37]. Initial research on video streaming focused on developing new streaming protocols for client-server video streaming on wired networks. As the popularity and number of users of video streaming continued to increase, P2P video streaming became popular. A wide varied research has been conducted in the last decade on the design of new P2P streaming protocols over wired networks [38].

To use P2P streaming, users were required to install dedicated applications. Furthermore, streaming traffic could be blocked by firewalls. These facts motivated the concept of streaming over the web using the HTTP protocol. HTTP video streaming is the most popular form of video streaming nowadays, and it has been adopted by major video streaming solutions such as YouTube and Netflix. This is due to the convenience of using the HTTP protocol [37], which eliminates the need to install and use a dedicated streaming application and helps to get the streaming traffic past firewalls. HTTP video streaming works by splitting the overall video stream into a sequence of small HTTP-based files, referred to as video segments. Users progressively download these small segments, while the video is being played. Playout usually starts after receiving a number of video segments. The received segments are buffered in a video/application buffer, as shown in Figure 2.2.

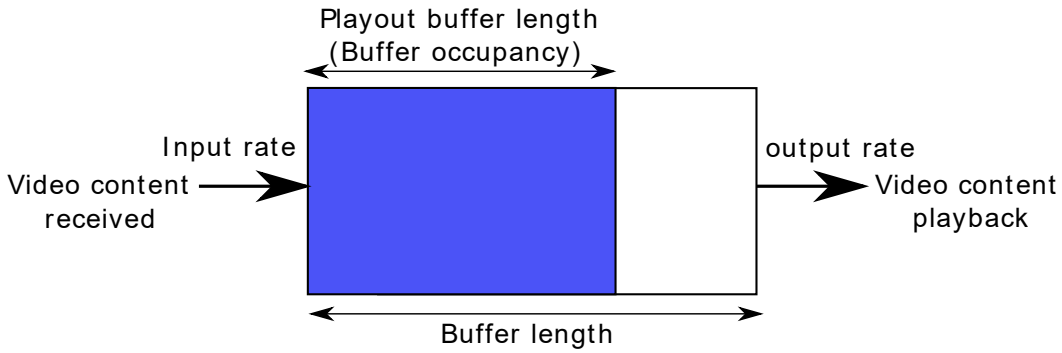

Figure 2.2. Operation of the video playback buffer.

The application that plays the video is usually referred to as the client. The streaming client requests video segments from the server. Since each segment has a fixed duration, the size of the segment depends on its duration and the video bit rate. The client receives the pieces from the video buffer. The duration of video content available for playout is

called the playout buffer length, measured in seconds of video. Furthermore, every second, one second of video is removed from the buffer and played to the user.

Bad network conditions (low bandwidth, long delay, etc.) may cause the playout buffer to get empty, as the video bit rate is higher than the video transmission rate, which causes video playout interruptions. These interruptions are referred to as video stalling or rebuffering. When stalling occurs, playout stops until video contents are buffered again.

2.1.3 Dynamic Adaptive Streaming over HTTP

Although HTTP video streaming provided a convenient way for video streaming over the internet, it was still challenging to stream video over networks with high bandwidth variability. DASH came as a promising solution to this issue [24]–[27], as it allows changing the quality of video streaming to adapt to network conditions. As we can see in Figure 2.3, DASH provides two features that helped improving video streaming. First, it breaks down the video into small, easy to download segments (e.g., 5-second chunks). Second, each segment is encoded at multiple bit rates. This provides multiple quality levels for each segment and allows adaptive streaming. Clients can choose between the various video bit rates to adapt to varying network conditions. When only a single video bit rate is available for clients as in classical HTTP streaming, this available bit rate could be lower than the transmission rate which leads to inefficient resource utilization due to the wasted extra bandwidth. This extra bandwidth could be used to download video contents with higher quality.

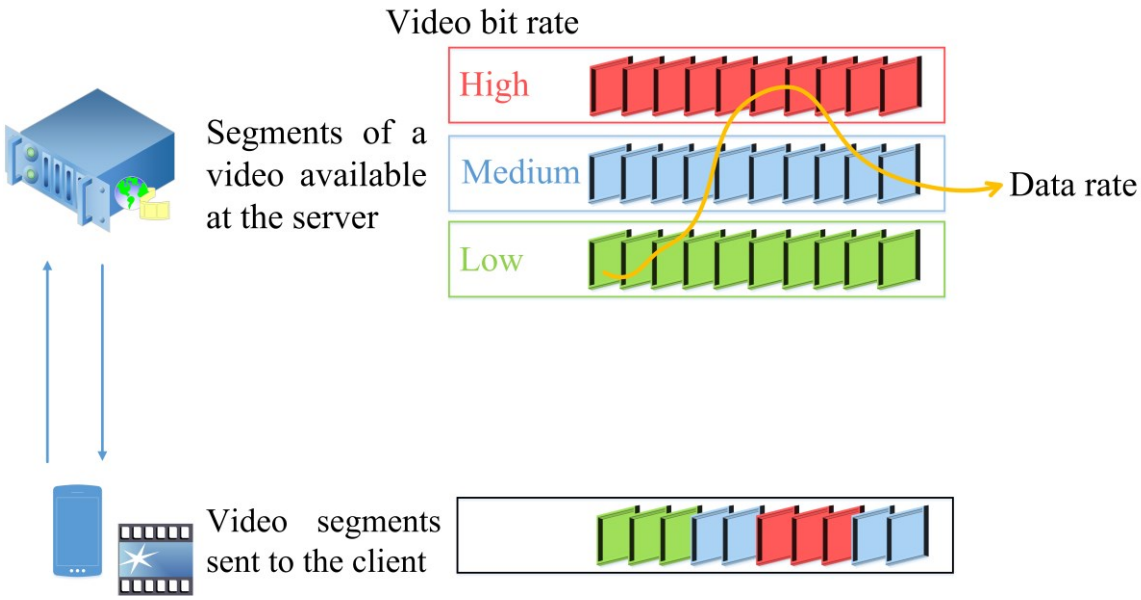

Figure 2.3. DASH operation.

On the other hand, if the video bit rate is higher than the available bandwidth, the video buffer will drain until it depletes, and video stalling will eventually occur which degrades the user's streaming experience. As such, DASH helps improving bandwidth utilization and reducing the interruptions of video playback, which results in a higher streaming quality. Due to these advantages of DASH over classical HTTP video streaming, DASH has been employed by big video streaming platforms, such as YouTube and Netflix, and it is being adopted by an increasing number of video applications.

In DASH, switching between different video bit rates takes place at fixed and frequent time intervals. As the video is divided into fixed-length small segments (e.g., 5-second segments), the adaptation controller in the client selects the video bit rate of the next segment to be downloaded. The adaptation strategy is the most important part of DASH on the client's side. It determines how the client selects the streaming quality to adapt to the varying network conditions. These strategies usually try to balance between two

factors. They try to maximize video quality by selecting the highest video bit rate the network can support, and at the same time avoid rebufferings.

Much work has been done on the adaptation strategies of DASH [39]–[43]. These can be classified into three categories: rate-based algorithms, buffer-based algorithms, and hybrid algorithms. In rate-based algorithms, the video client selects the video bit rate by monitoring network conditions and estimating the available network capacity from pervious observations. The DASH controller then would select the highest available video bit rate that is lower than the current throughput. The problem with these algorithms is that in environments with highly variable bandwidth, accurate estimation of future capacity could be challenging. Buffer-based algorithms were inspired by the fact that the occupancy of the playback buffer is the main variable the controller is trying to manage. Hence, video bit rate can be selected based only on the length of the playout buffer [41]. Hybrid approaches try to employ both the estimation of network capacity and the length of playback buffer. In these algorithms, the throughput estimation is used to select the video bit rate, but with an adjustment factor that depends on the current playout buffer length. The basic idea is that rate adaptation is done as per the capacity estimation, but the adaption follows the throughput estimation in a conservative manner when the playout buffer is small, and more aggressively with longer playout buffer. However, it is challenging to design an optimal adjustment functions that is aggressive enough to follow the available throughput and conservative enough to avoid video buffer depletion, especially under highly variable network conditions.

2.1.4 HTTP video streaming QoE

With the increasing demand for video applications, supporting high quality video services while achieving user satisfaction about the service has become an important concern for cellular network operators. This issue is becoming more important considering that not only the popularity of online video streaming is increasing, but also the quality of videos available via online streaming. For instance, YouTube and Vimeo nowadays provide 4K video support, which is a very high-resolution format with increasing demand [44]. At the same time, these platforms provide support for lower video qualities such as 240p or 360p.

Quality of Service (QoS) is measurement of the performance of a service and is usually evaluated in telecommunication networks by objective network parameters like the transmission rate. Never the less, providing good QoS to users in the network does not guarantee that all users in the network will have good experience with the requested services at the application level. This is becoming more common with the increasing popularity and quality of video streaming. For example, two users in the same cellular network might be watching two different videos, one with a low video bit rate and another with a high video bit rate (e.g., 4K video). In such case, providing both users with the same data rate (QoS) might cause the user with the low video bit rate to enjoy a smooth video stream and the other, with the high video bit rate, to experience many rebufferings, and hence, low QoE. Even if both users are provided with a decent data rate, the provided rate could still be lower than the higher video bit rate, which results in multiple rebufferings. Consequently, quality measure has shifted from QoS to QoE [45], [46]. The ITU defines QoE as the overall acceptability of the application or service as

perceived by the end user [47], [48]. It is a measure of the level of satisfaction or annoyance of the user with the overall service. For instance, a user who is watching a YouTube video over a cellular network is oblivious to QoS metrics such as the data rate. In such case, user satisfaction (QoE) is directly impacted by how pleasant the video streaming experience was, i.e., the quality of the video, number of playout pauses, etc. Video streaming QoE is very important because users pay their operator and they expect support for video services with good QoE in return. If the user is not satisfied, they may look for other options and switch to another provider. As such, video streaming QoE should be considered in network design and management in order to maintain user satisfaction.

Although video streaming QoE is concerned with the quality as perceived subjectively by the end user, many studies have been conducted to determine the objective metrics that influence users' opinion and decide video streaming QoE [46], [49]–[53]. These metrics provide objective way to study, evaluate, and improve video streaming QoE. Regarding HTTP video streaming, it has been shown by many studies that video rebuffering and initial delay are very important factors on the end-user QoE [46], [49]–[53]. The quality of the video measured by the video bit rate is also an important factor on the end-user QoE of HTTP video streaming.

Video stalling (rebuffering) is the interruption of video playback as the playout buffer gets empty. As has been shown in Figure 2.2, this happens when the video bit rate is higher than the transmission rate. When this happens, video playback stops until a certain amount of video content is buffered again. The duration of video buffered should be short enough to avoid long rebuffering delay, and long enough to avoid potential video buffer

depletion. As such, the duration of buffered video provides a tradeoff between rebuffering time and number of rebufferings during video playout. Obviously, video stalling decreases video streaming QoE. The authors in [52] have shown that although increasing rebuffering time has a negative effect on the QoE, users usually prefer one long rebuffering duration over multiple shorter rebufferings. Many studies [51], [53] have shown that video stalling has the biggest impact on QoE, and even few short rebufferings could have severe impact on the QoE. As such, video stalling should be avoided as much as possible.

Video continuity index, $\eta_c \in [0,1]$, is a metric that combines the number of rebufferings and rebuffering time [42]. It is a measure of the extent by which rebuffering pauses are avoided [42]. The continuity index is measured as follows,

$$\eta_c = 1 - \frac{\Delta T_{rb}}{\Delta T}, \tag{2.1}$$

where ΔT_{rb} is the total time the client remains paused due to rebuffering events and ΔT is the duration of the experiment (playing time and rebuffering time). When zero rebufferings occur during playout, rebuffering time will be zero, which gives a continuity index of 1. This is the best achievable value for the continuity index.

Initial (startup) delay is the delay from the request of the video stream to the actual start of the video stream. Initial delay is always present as certain number of video segments should be received before decoding and playback start. As such, it depends on the available throughput. The initial delay is usually extended so that the playout buffer at the client builds up before playout starts in order to avoid video buffer depletion that could

be caused later by throughput variation. As such, the duration of video buffered at the beginning should be short enough to avoid long initial delay, and long enough to avoid potential video buffer depletion.

Although initial delay has an impact on video streaming QoE, studies have shown that it has less effect than video stalling [51], [53]. It has been noticed by [51] that initial delay is preferred over rebuffering most of the time. This is mainly because initial delay is usually expected, especially for relatively longer videos. In many cases, users actually anticipate initial delay and even pause the video right from the beginning so that video contents are buffered. Video stalling, on the other hand, is a sudden unexpected interruption in video playback that is found to be more annoying by most users, and hence, causes more degradation to the QoE. Due to the results reported by such studies, initial delay should be kept short, but it is considered less critical when it comes to achieving good QoE, and initial delays up to several seconds can be still tolerated by users.

Video bit rate is a measurement of the amount of data in one second of the video. Video bit rate is determined by many quality factors of the video such as video frame rate, resolution, and quantization parameters. As the video bit rate increases, the video quality increases, which increases the QoE. It is usually measured in terms of the average video bit rate (average playback bit rate), which is the average video bit rate of all the video segments in the video stream.

In our research, we consider all the factors above when evaluating video streaming QoE. To objectively evaluate all the factors above, we consider the following QoE metrics,

- Number of rebufferings

- Continuality index
- Initial delay
- Average video bit rate

2.1.5 Discrete EVent System Specifications

We built various DEVS models throughout this work to evaluate video transmission data rates as well as video streaming QoE over cellular networks. In this section, we provide an overview on DEVS.

DEVS provides a formal framework for modeling generic dynamic systems [21]. It has formal specifications for defining the structure and behavior of a discrete event model. A DEVS model is composed of structural (Coupled) and behavioral (Atomic) components, in which the coupled component maintains the hierarchical structure of the system, while each atomic component represents the behavior of a part of the system.

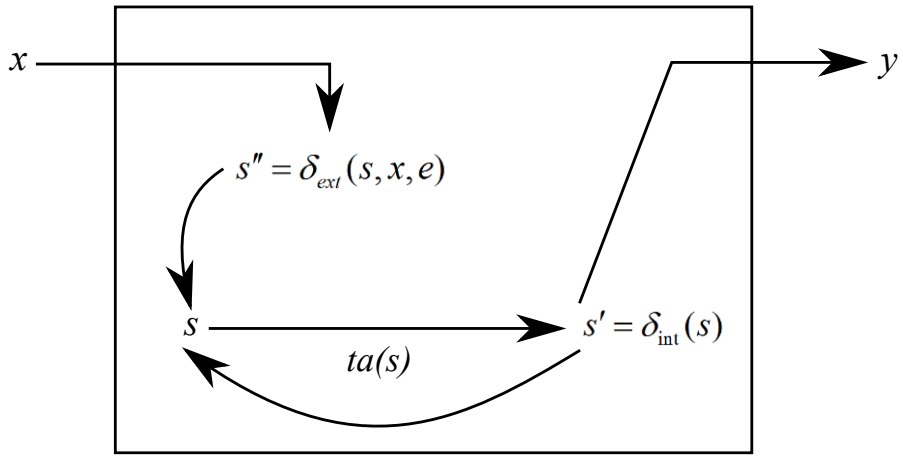

Figure 2.4. DEVS atomic model (adopted from [21]**).**

Figure 2.4 depicts a DEVS atomic model. The atomic component uses I/O ports and a finite state timed automaton representing the behavior of the model. A model is in state *s*

for a specified time *ta(s)*, after which it produces an output *y* and changes its state based on the internal transition function, $\delta_{\text{int}}(s)$; a transition that takes place due to the consumption of *ta(s)*. If it receives an input *x* before *ta(s)*, it invokes its external transition function, $\delta_{ext}(e,s,x)$, which can also change the model's state, where *e* is time elapsed since the last transition.

A DEVS coupled model contains multiple atomic or coupled models that are connected through I/O ports. Figure 2.5 shows an example of a coupled model that contains three atomic models. The first and third atomic models are connected to the external coupled model through external input and output couplings, respectively. Coupled models group several DEVS into a composite DEVS model, allowing hierarchical construction.

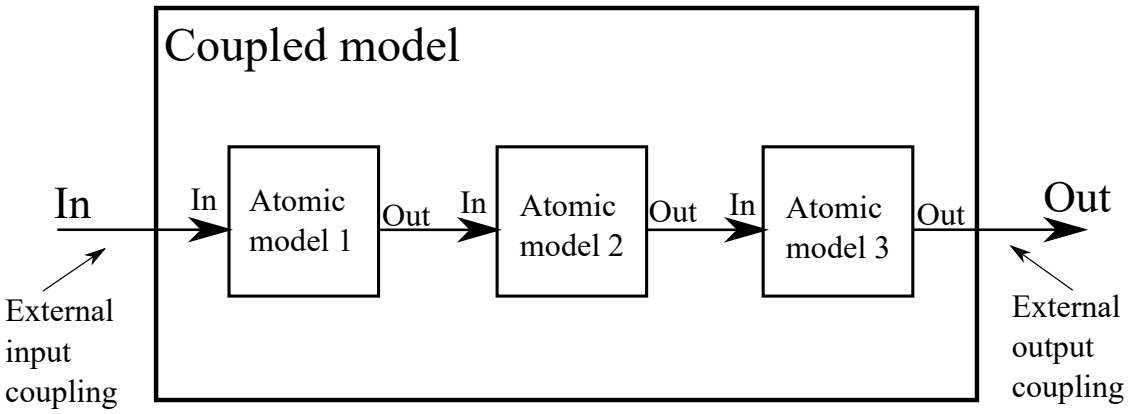

Figure 2.5. A DEVS coupled model.

This modular and hierarchical nature is a very useful property for modeling and simulating LTE-A networks. The network model can be built using different submodels; each one implements a different component of the wireless network such as the BS or the UE. Each one of these submodels can be tested and verified independently, and integrated into the whole model. These submodels can also be reused in other LTE-A

network models. Furthermore, each submodel, such as the BS, can also be implemented using multiple submodels in a hierarchical manner. Each one of these submodels implements a certain subcomponent or functionality. This makes it easy to design, implement, and test LTE-A network models.

The CD++ toolkit [21] was used to implement our DEVS models. CD++ is an open-source simulation software written in C++ that implements the DEVS abstract simulation technique. The simulation engine tool of CD++ is built as a class hierarchy [21]. C++ is used to develop the atomic components of the model. These components can be incorporated into the class hierarchy. Passive classes can be also used to model components of the system. Coupled models can be created using a language built in the simulation engine.

2.2 Related work

In this section, we discuss the related work in the literature for each one of the topics involved in this thesis.

2.2.1 Video caching and D2D video content delivery

Nowadays, there is an enormous amount of data that is available on the Internet and requested by users. As per [22], the annual global IP traffic is expected to reach 3.3 zettabyte by 2021 (a zettabyte equals one trillion gigabytes). That is a huge increase from the annual global IP traffic in 2016 which was 1.2 zettabyte at the time. Furthermore, video is expected to account for 82 percent of the global IP traffic by 2021 [22]. Due to this dramatic increase in the amount of data traffic (particularly video traffic) to be distributed to users, new proposals and solutions have been developed to improve content

distribution to end users over the Internet. Content Distribution Networks (CDNs) is a solution that enhances the delivery of contents to users by making them available over a geographically distributed network of servers. Information-Centric Networking (ICN) is another solution that was proposed to improve Internet content delivery [54], [55]. Currently, a host-centric approach is used for communication over the internet. With host-centric communication, a user needs to provide, with every data/information request, the address of the server/host that contains the data. ICN is motivated by the fact that nowadays, users usually want to access data/information regardless of its location or the host that provides it. This makes the currently employed host-centric communication approach inefficient. The basic idea of ICN is that each content is assigned a globally unique name, and information is addressed and identified using its name as apposed to its location or host [55]. This approach leverages in-network caching. When a user wants to access a content (data object) on the web, they send a request that contains the name of that content, and the network forwards the request (based on the content's name) to the best node in the network that has that content. Although ICN presents a promising solution for efficient and reliable distribution of content on the Internet, it proposes a drastic change to the current architecture of the Internet. ICN still has many challenges such as scalability and deployment, and all the current proposals are under development [54], [55]. While the above solutions do improve distribution of web contents in general, they do not solve or target the problems of cellular networks such as the huge amounts of video traffic on the backhaul or the RAN bottleneck of cellular networks. As such, there has been an urgent need for solutions for cellular networks' problems, and more specifically, video delivery over cellular networks, as it accounts for the majority of data traffic.

In [56], the authors proposed caching on the end-user side video contents that are expected to be requested in the future. An adaptive popularity-based video caching strategy can be employed. The strategy enables strong collaboration between users and the service environment for ensuring better QoS to end users. Video contents are dynamically cached in home-boxes following users' demands, allowing their delivery from optimal places. Although such approach helps improving the delivery of video traffic for some users, cached contents can be only used locally by the caching devices, and cannot be exploited by others in the network.

Caching contents at the BSs has been employed to help improve the transmission of video traffic in cellular networks [57]. When a popular file is cached at the BS, it will be available (at the BS) when requested by the UEs, which eliminates the need for requesting the video from the content server and reduces the amount of traffic on the backhaul network. In [58], an information-centric solution was proposed to improve video traffic delivery over cellular networks. In this solution, a video-centric proxy cache for mobile networks is employed. This proxy cache, which is called iProxy, can be implemented at or behind the BS in cellular networks. The proposed iProxy utilizes the information-bound reference of requested videos. The information-bound reference of a video provides information on the frequency domain representation of the video data. As such, it is tied to the information in the video rather than the host, filename, bit rate, or resolution of the video [58]. iProxy extracts the information-bound reference for requested videos and keeps a table of the URLs of requested videos and their corresponding information-bound reference. Because the information-bound reference represents the information in the video rather than its URL or quality, different URLs can

map to a single information-bound reference. When iProxy receives a video request, it checks if the URL is available in its table and if it maps to an existing information-bound reference. By using such information-centric approach, the hit ratio for the utilized cache increases which reduces the delay to deliver the requested videos. The solutions in [57], [58] do not reduce the amount of traffic between the BSs and UEs over cellular frequency links which is the main bottleneck in cellular networks with high user density. Although deployment of small cells might further leverage video caching in micro BSs and improve frequency reuse, it still does not reduce the amount of traffic between the BSs and UEs over cellular frequency links. Furthermore, such infrastructure with small cells might not always be a viable option because it becomes more difficult to handle mobility and control interference when the size of the cells decreases, and it is costlier to implement a high-capacity wired backbone as the number of cells increases.

Caching video content on servers close to the users combined with coded-multicasting was proposed in [59]. The authors propose employing simultaneous coded-multicasting by the caching server to satisfy the requests of several users with different demands in a single multicast stream. Nevertheless, such scheme requires complex and sophisticated methods to have contents available a priori at both the caching servers and requesting devices. Because contents placement should be done before users' demands are known, it must be designed carefully such that these coded-multicasting opportunities are available simultaneously for all possible requests.

Despite the improvements achieved by the approaches above, the increasing number of users and demand for video contents in cellular networks make it challenging to serve video contents with such client-server approaches. New and more scalable architectures

are needed for video content distribution in cellular networks with high number of users and limited resources. As such, there has been a need for P2P communication models at the user level. There has been some work on P2P communication in wireless ad hoc networks [60]. With cellular networks, on the other hand, P2P communication started getting popular after the emergence of D2D communications [8], [9], as it allows direct communications between UEs in cellular networks. In [34]–[36], the authors propose many incentive mechanisms to motivate the involvement of UEs in D2D communication. This is important because the success of D2D communication depends on the participation of users to share the contents they have.

Some of the work in the literature is focused on caching contents in smart devices. In [61], mobile content delivery networks were proposed where special mobile devices are placed in the network as caching servers. Popular contents are cached in such servers and can be provided to users in their vicinity over D2D communication. While this technology could help improving the data rates in cellular networks, it is costly as these designated devices need to be placed throughout the network, configured, and maintained. The architecture, proposed in [17], [62], employs D2D communication to improve the throughput of video transmission and overcome the problem of rapidly increasing wireless video traffic. In this architecture, UEs in the cell can save video files. When a video file is requested by a UE, it will be transmitted from a close UE that has the file over a D2D link, if the file is cached. The network model in [17] is oversimplified and the original architecture is limited; it assumes that the files are pre-cached in the nodes. The work also assumes that complete files (as apposed to segments) are cached and exchanged between the network nodes. Furthermore, they do not define a messaging

protocol between the UEs and BS to exchange such video files. Instead, a simple model was used to study the performance of the architecture analytically.

The work above either focus on caching polices for video/data contents assuming that caching of such contents takes place in designated devices and during low traffic load, or focus on the impact of D2D communication assuming that data is already cached. Under high traffic load, it would be highly inefficient to transmit unrequested data/video contents for caching under the assumption they might be requested in the future as this would waist the precious cellular resources. Furthermore, the popularity of videos might not be known for the network and need to be "learned" progressively. Moreover, mobile devices are usually not stationary for extended periods of times to be treated as caching servers (content is solely transmitted for future utilization). As such, in [12], [14]–[16], we propose our cached and segmented video download algorithms for BS-assisted progressive caching of video segments in UEs and P2P transmission of video contents among UEs using D2D communication. The proposed algorithms, namely CSVD and DISCS, are presented in Chapter 3. The algorithms employ an improved architecture of the one in [17]. Only selected UEs in each cluster are used for caching to reduce inter-cluster and inter-cell interference between D2D transmitters. Instead of caching complete files, the files are divided into segments and multiple copies of a file can be cached at multiple SMs. SMs are assigned to requesting UEs on a segment-by-segment basis (rather than permeant pairing), depending on many factors, such as the availability of segments and SMs. Furthermore, the algorithms employ new technologies in the 5G cellular networks such as multi-RATs, dual connectivity, and flow splitting/aggregation [18] to send multiple video segments to a user (from more than one source) simultaneously, when

possible. A protocol has been defined for the algorithms, including a variety of messages necessary for this communication, and a complete definition of the protocol is described. The operation of the algorithms in the case of UE mobility is also considered.

The proposed algorithms do not consider only the transmission of video segments to requesting UEs over D2D links, but also caching of video segments. Locally-popular videos can be strategically cached to be available to all users in the cell. Furthermore, the proposed algorithms can be used to progressively cache video segments as requested, and do not assume that all videos can be always cached well in advance. Although these characteristics, of the proposed algorithms, are beneficial in any video streaming scenario with high number of users, they are crucial for high traffic load scenarios where users are requesting recently published contents. Such contents are transmitted to requesting users over cellular resources. As such, under high traffic load, it would be highly inefficient to transmit unrequested video contents for caching under the assumption they might be requested in the future, as this would waist the precious cellular resources. At the same time, requested video segments (especially the locally-popular ones) should be cached in case they are requested later. Such scenario is common nowadays because many video streaming websites, including YouTube, provide a service that is a combination of live and on-demand video streaming. In such cases, many users are watching the video stream around the time it is published. However, viewers do not have to be synchronized, i.e., some users might be watching the beginning of the stream while others could be watching parts seconds or minutes later in the stream. This is a motivation for progressive caching, where users who started watching early can provide recently published contents to others.

In CSVD, the cell is divided into clusters. Selected UEs in each cluster are chosen as SMs. When an SM requests a video file, the BS will send the segments of the video file to the SM over a cellular link. The SM will be requested to cache the downloaded video segments. When cached video segments are requested by other UEs in the cluster, the segments will be sent over D2D links from caching SMs. In DISCS, a popular video file is divided into segments. The segments of the video file are distributed over multiple SMs to be cached and forwarded to the requesting UE. This provides further parallelism when transmitting video files and further load balancing among SMs, which speeds up the transmission process. Furthermore, in DISCS, the SMs might be required to receive and cache segments that are requested by others when asked for assistance (as opposed to just forwarding pieces they already have) which helps accumulating video files faster in the distributed cache.

2.2.2 Evaluation of video streaming over cellular networks

As previously mentioned, the parameters of the cellular transmission over which video segments are delivered to UEs have an impact on the end-user QoE. There has been some work studying the effect of the network conditions on the QoE in wired networks. In [63], the authors studied the correlation between the network QoS such as delay and throughput, and the QoE of HTTP video streaming in wired networks.

There is also some work in the literature on the QoE of video streaming over cellular networks. In [64], the authors presented an Android application that passively monitors key performance indicators of YouTube adaptive video streaming on smartphones. The indicators include player's state/events, buffer length, and video quality level. These could be used to analyze the QoE of mobile YouTube video sessions. The application

was tested through real subjective QoE tests showing that the tool accurately captures the experience of end users watching YouTube on smart phones.

In [65], the authors presented an Android application that can evaluate the perceived QoE of YouTube in wireless terminals. The application makes objective QoS measurements that are then mapped onto subjective QoE (in terms of Mean Opinion Score (MOS)) using a utility function. The application reports to the user the potential causes that might have led to a low MOS, and it provides suggestions for improvement. Furthermore, it allows users to rate each session through an online opinion survey.

In [66], the authors studied the influence of the wireless network conditions such as SNR, fading, and latency on the quality as perceived by the end users. QoE was assessed in terms of the MOS based on subjective measurements. Different scenarios with different network conditions were considered. In each scenario, a set of video segments (called the HTTP adaptive streaming profile) was created. Volunteers then were recruited to watch the resulting HTTP adaptive streaming profiles and rate each one to generate a MOS value.

In [67], the performance of YouTube flows accessed through a cellular network was analyzed in terms of the download throughput as well as the end-user QoE. The analysis considers the influence of the content delivery network hosting YouTube. The authors in [68] studied the problem of QoE provisioning for popular applications in smart phones. They study the effect of the access DownLink (DL) bandwidth on the QoE of popular applications such as YouTube and Facebook.

Here, we go a step further and study the effect of the parameters that control the cellular

link conditions on the end-user QoE. We consider parameters at both the cell level (cell bandwidth and number of UEs in the cell) as well as the link level (BS transmission power to the UE and distance between the BS and the UE). We consider various scenarios including ones where the cell is overloaded. To the best of our knowledge, this is the first work that considers the correlation of cellular transmission parameters, at both the cell level and link level, to video streaming QoE. Such study is important because the operators are changing their perspective from network-centric to experience-centric operation. As video traffic is dominating mobile data traffic, providing good QoE video service is a key to differentiated competitiveness. As such, this study provides a valuable insight for LTE-A network design, especially in terms of small cells deployment and configuration in urban areas (where there is usually a high density of users).

2.2.3 D2D video streaming over cellular networks

There is some work in the literature on D2D video streaming in cellular networks. In [69], a system, called MicroCast, was designed and evaluated using a testbed. MicroCast is used by a group of smart phone users who trust each other, are interested in watching the same video at the same time, and who are within proximity of each other. Users use their cellular connection to download segments of the video and use their WiFi connections to share among each other the downloaded content to improve user's experience.

A similar P2P application was developed in [70] for live streaming of video contents. The application is designed for a small set of devices that have both cellular and WiFi connections and interested in watching the same live stream. Periodically, users randomly select chunks of the live stream to downloaded through the cellular connection. Then,

they share among themselves the downloaded chunks to improve the playback quality of the live video stream. The application in [70] exploits some of the functionalities of ICN such as unique hierarchical naming and routing by name. These functionalities are used by the participating devices to announce the chunks they have and to exchange chunks among themselves. While the proposals in [69], [70] could result in some improvement for a group of users, their scope is limited as they are designed for a small group of users. Furthermore, these systems are designed for live video streaming where users have synchronous playout of the video.

It is worth mentioning that with ICN, each data object should have a globally unique name, and these names are used to route data requests from receivers to an available source. Implementing routing by name on the Internet is complicated and have many challenges. These functionalities of existing ICN systems such as naming and routing by name can be employed with D2D content sharing as in [70]. However, for D2D content sharing among users in the cell, any system that keeps track of the available contents (e.g., URLs) and caching devices can be used (especially with a network-assisted approach).

A protocol for P2P video streaming on mobile phones, called RapidStream, was proposed in [71]. It is similar to many of the P2P streaming protocols on wired networks that involve the dissemination of buffer maps and video chunks between peers. While such protocols work well in wired networks, they involve too much signaling and transmission (e.g., dissemination of buffer maps) to be scalable for UEs that has limited power, processing, and transmission resources. In [72], multi-source video streaming was proposed where mobile users can connect through WiFi direct to other users to get some of the video contents. Such system requires the device to perform device discovery to

find neighbors, and service discovery to find services offered by neighboring device. As discussed in some studies [7], such service and peer discovery requirements along with the signaling needed to exchange contents consume significant amount of resources from the UEs.

In [73], the authors proposed a D2D communication system where multiple helpers collaborate to send video segments to a requesting UE. The video is encoded by applying multiple description coding by each helper. Each helper sends a different description to the requesting UE. The authors analytically studied the problem of optimizing the number of transmitted descriptions to the requesting UE to maximize video quality and efficiently consume the helpers' energy. However, the work in [73] does not consider the most important metrics of video streaming QoE such as the number of rebufferings and initial delay. As discussed above, improving video bit rate/video quality increases the end-user QoE. Never the less, increasing video bit rate could increase transmission delays which increases the number of rebufferings and initial delays. Such factors are very important to consider as degradation of these metrics would significantly worsen video streaming QoE. Furthermore, the work only considers the energy consumed by the helpers to send the segments without considering the processing power and energy needed to encode the video segments. Encoding the video segments is a big favor to ask for, considering the limited energy and processing power of UEs. Moreover, the optimization problem assumes that the BS knows the energy levels of all helpers.

None of the work in [71]–[73] on P2P video streaming in cellular networks considers how video segments are actually cached. When evaluating the performance, they consider that requested segments are pre-cached. Furthermore, all the work above

considers small-scale networks, i.e., up to 10 UEs including the helpers. The number of UEs per cell in urban areas are usually much more than that. Furthermore, in some scenarios, discussed in Chapter 3, the cell could be very condensed. We show that employing CSVD and DISCS by the BS achieves significant video streaming improvements in high user density scenarios.

In Chapter 7, we propose and evaluate DABAST; an architecture to improve the QoE of video streaming in cellular networks with high user density. DABAST employs BS-assisted progressive caching and D2D video transmission between UEs (CSVD and DISCS), as well as DASH. Both DASH and the proposed algorithms (CSVD and DISCS) are very beneficial for video streaming over cellular networks. Combining both techniques in one system is a necessary step that brings many benefits and achieves performance gains to video streaming on cellular networks in high traffic load scenarios. However, such system needs much study. With DABAST, we consider the case of progressive caching of video segments under high traffic load. As discussed above, this is motivated by common scenarios where many UEs could be requesting a recently published popular content under high traffic load. We consider the case where there is an overloaded and limited cellular channel that provides access to any video segment at any available video bit rate, and an out-of-band D2D channel with much higher bandwidth that only provides access to certain videos (at certain video bit rates). DABAST provides a system that makes the best of available channels and cached segments. With DABAST, SMs are assigned to requesting UEs on a segment-by-segment basis, and we optimize this assignment to maximize the QoE of end users. Furthermore, we introduce QoE awareness to video caching. From all the above, one can see that many decisions need to be taken

dynamically regarding caching and distribution of video contents, as well as allocation of resources and SMs to improve video streaming QoE for users in the cell. Another feature of DABAST is that it utilizes new technologies in the 5G cellular networks such as multi-RATs, dual connectivity, and flow splitting/aggregation [18], [19] to send multiple video segments to a user (from more than one source) simultaneously, under some conditions. These features are utilized by DABAST in the "proactive mode". The proactive mode is discussed in Chapter 7.

The work in [74] considers scheduling and rate adaptation in wireless networks with multiple helpers. However, it considers the case where helpers are connected to a wired backhaul (e.g., small BSs) and have the videos in all available quality levels. In DABAST, helpers are UEs that are caching video contents progressively as requested (over the limited cellular channel), and that UEs request from a high number of files. As such, not all requested videos are available in the distributed cache and an SM would have a segment (if available) in one quality level. Hence, the availability of segments/quality levels in SMs are taken into consideration in the operation of DABAST such as SM assignment. Furthermore, the work does not consider interference management and that multiple helpers can transmit at the same channel. With DABAST, interference between SMs is controlled through clustering, selection of SMs, and radio resource management by the BS. Unlike the work in [74], we do not consider that requests of video segments coincide with scheduling/assignment time as we consider users request at random times (as in a real scenario). Moreover, in our work, we consider the case of high user density and high traffic load, and hence, clustering and BS control are utilized to manage the different aspects of communication between the high number

of users (such as interference, scheduling, and SM assignment). Furthermore, DABAST is designed to increase the utilization of cellular resources, in such scenarios, as they are the only way to access segments that are not available in the distributed cache. Finally, DABAST can operate in proactive mode where more than one segment can be sent to a UE (from different sources), if needed, which further reduces rebufferings and initial delays.

In [75], the authors proposed two scheduling algorithms for the delivery phase in D2D video streaming. The algorithms allocate frequency resources to several D2D links, where each link is between a pre-matched pair of a helper (provider of video content) and a requesting device in the network. A similar algorithm has been also proposed in [76], where a single channel is shared between multiple D2D links between pre-matched helper-receiver pairs. As with the previous work above, the work in [75], [76] assume that all videos are pre-cached in the user devices before video streams start, and that each caching device already have the video available at multiple video qualities. Our proposed architecture considers not only the delivery phase, but also video caching in the UEs in a high traffic load scenario. Furthermore, the work in [75], [76] only considers the D2D communication between the UEs in the network over a single channel, considering that all requested videos are cached. DABAST, on the other hand, provides a framework that considers not only the D2D communication between UEs (on a D2D channel), but also the cellular communication between the UEs and the BS. This is crucial because in a real scenario, many of the requested videos might not be available in the distributed cache in the cell, and hence, should be downloaded over the cellular channel through the BS. Moreover, the work in [75], [76] assumes that a UE is only provided the segments of a

video by one helper. In DABAST, SMs are assigned dynamically to requesting UEs on a segment-by-segment basis, depending on SMs availability and video bit rates of available segments to maximize video streaming QoE.

2.2.4 Video streaming QoE-aware resource management in cellular networks

The continuously increasing demand for HTTP video streaming over cellular networks made it very important for network operators to support good QoE video streaming services. Many scheduling algorithms have been proposed for cellular networks based on QoS objectives in the network such as fairness in allocating resources or maximizing the network throughput. However, providing a certain QoS level to users in the network does not guarantee that all users can get video streaming service with good QoE, due to the complex and dynamic nature of video contents. For example, two users in the network can be watching videos with different video bit rates. Providing a decent and equal data rate to both users does not guarantee that both users will have good QoE, as the user with the higher video bit rate (e.g., 4K video) might experience higher delays and more rebufferings. The situation is more complicated considering that the used video bit rate might be varying, as in DASH, and that some clients might be buffering or pausing, while others are watching the requested videos. As such, there is a need for new resource allocation approaches to maximize video streaming quality as perceived by the end users.

Many video streaming QoE-aware approaches for resource allocation in conventional cellular networks (without D2D communication) have been proposed [77]–[82]. In [78], a DASH video steaming QoE maximization approach was proposed for resource allocation in LTE networks, where the playout buffer levels are signaled from the UEs to the QoE optimizer at the BS. The optimizer considers the playout buffer levels at the UEs

to perform a multi-user resource allocation. The playout buffer levels are also considered for multi-user video bit rate adaptation at the BS. A similar approach that employs the playout buffer length is proposed in [79].

Other approaches use a utility function that maps technical parameters such as transmission delay to QoE scores. Based on this mapping, resource allocation is performed to guarantee certain QoE score to users. In [80], static learning is used to build a mapping function between QoE scores and technical parameters. This function is used in resource allocation to provide certain QoE levels. A similar approach is used in [81] where random Neural Networks (NNs) are used to build a utility function to be used in resource allocation. In [82], a statistical model is used for mapping between network parameters and QoE sores, and this statistical mapping is used for resource allocation. The problem with the methods that depend on utility functions for mapping is that such models do not provide an accurate and real time evaluation of users' QoE, especially considering the complex and variable characteristics of video contents as well as the video playout dynamics at the clients such as video playout and rebuffering.

The work in [75], [76], adopt QoE-aware approaches. They utilize the queue backlog size at the senders or an estimate of the playout buffer as the metric to avoid causing long delays. However, the queue backlog size at the senders or the duration of transmitted video contents do not provide an accurate and real time way of tracking the playout buffer at the clients considering the video playout dynamics at the clients (playing, rebuffering, or pausing) and that a user might get video contents from different sources.

In Chapter 8 and Chapter 9, we employ QoE awareness in three aspects of DABAST, namely, cellular resource allocation, caching of video segments, and SM assignment optimization. For cellular resource allocation, we propose a scheduling metric that employs the playout buffer length which is similar to the one in [78] for traditional cellular networks (without D2D communication). Our scheduling algorithm is proposed to operate in DABAST and considers not only the reported playout buffer length, but also an estimation of the transmitted video contents (since the last update) over both the cellular and the D2D channel (from all sources). Furthermore, we propose an approach for high video bit rate caching of segments in the UEs that also utilizes the metric above which keeps track of the playout buffer length at the clients. Finally, we propose an approach for SM assignment optimization; to dynamically and adaptively maximize the achieved video bit rate for users in the cell, while minimizing rebufferings. Results show that all the QoE awareness techniques above do indeed improve the performance gains achieved by DABAST.

Chapter 3

The CSVD and DISCS algorithms

In this chapter, we start by presenting the proposed algorithms and how they operate. Afterwards, we provide analytical models for video caching with both algorithms. Closed-from expressions are derived for the efficiency of both algorithms in terms of the hit ratio. Thereafter, we discuss the operation of the algorithms under UE mobility and various implementation issues.

3.1 Operation of the proposed algorithms

Both CSVD and DISCS are designed for scenarios where there is a high density of users in the cell, such as:

- Sport events in which users want to download instant replays from this event, or videos of other events taking place at the same time
- Live concerts with detailed video feeds of the arena
- Massive religious events (e.g., a Pope's Mass in St. Peter's Basilica in the Vatican)
- Large political events (e.g., election results or inauguration speeches)
- University convocations

Consider one cell in a cellular network, in which the BS is in the center of the coverage area, as seen in Figure 3.1. In this analysis, we consider the case where UEs are stationary. Later in this chapter, we discuss the operation of the algorithms under UE movement. At the beginning, the BS starts by dividing the cell into clusters, as follows:

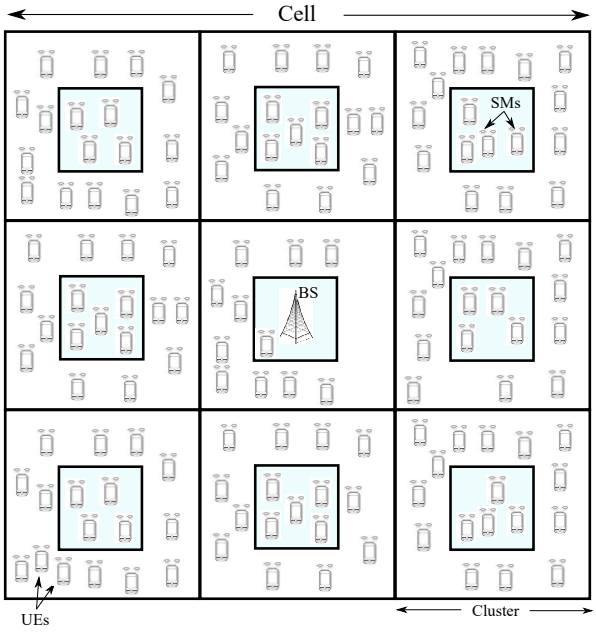

Figure 3.1. A cell divided into 9 clusters.

1) The BS logically divides the coverage area into non-overlapping subareas. Each one of these will be a cluster.
2) The BS sends a broadcast *Clustering* message telling the UEs that a cluster formation is about to start.
3) The UEs reply with a *Clustering Response* message indicating their location.
4) The BS assigns UEs to clusters based on their locations, and it selects the UEs in the central area of each cluster as the SMs of that cluster, as in Figure 3.1. Only the UEs in the central area of each cluster are chosen as the SMs to prevent inter-cluster and inter-cell interference when the SMs transmit to other UEs in the same cluster using D2D links.

After clustering, the transmission phase starts. The UEs send their requests to download video files to the BS. The BS processes each download request, and responds differently to each request depending on the case. We consider four different cases. Three of the

cases below apply to both CSVD and DISCS, and one of the cases is only used in DISCS. The cases are as follows:

- Send With Assistance (SWA): if the video file (or a part of it) is available in any of the SMs of the cluster, the BS will ask these SMs to send the pieces they have to the requesting UE over D2D links.
- Send To an SM (STSM): if the requested file is not available in the distributed cache (or more copies need to be cached), and the requesting UE is an SM, the BS will send the file to that SM over a cellular link, and ask the SM to cache the video file. Cached video files will be available for UEs in the cluster when requested later.
- Distribute To SMs (DTSMs): this case is only used in DISCS. If a requested video is popular (requested *n* times) and it is not available in the distributed cache of the cluster, the BS will distribute the pieces among the SMs in the cluster. The BS asks the SMs to cache the pieces (as the file is popular), and asks them to forward the received pieces to the requesting UE.
- Send To a UE (STUE): otherwise, the BS will send the file directly to the requesting UE over a cellular link.

In the following sections, we discuss the different cases described above in detail.

3.1.1 Send with assistance

In this case, the download process, which is shown in Figure 3.2, takes place as follows:

1) The UE sends a *Download Request* message to the BS.
2) As this file has already been sent before to an SM to cache it, the BS has a *MetaInfo*

file that describes the parameters for the download session of this video file. Table 3.1 shows the fields of the *MetaInfo* file. The fields in the *MetaInfo* file represent the parameters for this download session. The BS then sends a *Handshake* message to the requesting UE. The *Handshake* message contains the *MetaInfo* file.

3) The BS will check its database to find out which of the SMs have the pieces of the cached file. Then, the BS will send an *Assistance Request* message to these SMs asking for their assistance to send pieces to the requesting UE. The *Assistance Request* message has a field indicating the number and indices of the pieces that the SM should send to the requesting UE.

Table 3.1.MetaInfo file.

Field	Description
File Size	The file size in bytes
Number of Pieces	The number of pieces
Piece size	The piece size in bytes
Last piece size	Last piece size in bytes
File name	A string representation of the file
Info	A dictionary that describes the file

4) The SMs will send a *Response* message. The SMs will indicate whether they are available to assist with this download session or not. The Response message also contains a field that indicates the maximum number of outstanding assists the BS should send, i.e., the maximum number of assists this SM can handle at a time.

5) The BS and the SMs start sending the pieces to the requesting UE. Each time the BS wants an SM to forward new piece(s) of the video file, it will send that SM an *Assistance Request* message indicating the piece(s) to forward. Each piece message has an index that identifies that piece.

6) When an SM finishes sending piece(s), it will send an *SM_Finished* message to the BS, acknowledging the transmission of the piece(s).

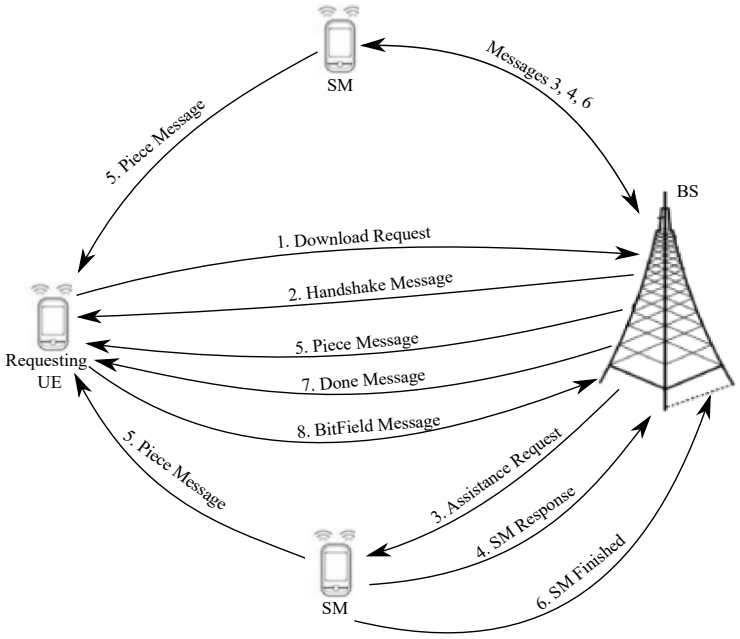

Figure 3.2. SWA case.

7) When the BS receives *SM_Finished* for the pieces from the SMs participating, and when it finishes sending its pieces, it will send a *Done* message to the requesting UE.
8) When the requesting UE receives a *Done* message, it will send a *BitField* message to the BS indicating the pieces it has received.

The BS should keep in its database the following:

- A list of the clusters
- A list of the members and SMs of each cluster
- A list of the cached video files/segments, and caching nodes

3.1.2 Send to an SM

In this case, the file transfer starts by the SM sending a *Download Request* message to

download a video file. After receiving the request, the BS will start a session with this UE. If this is the first time an SM requests this file, the BS creates a *MetaInfo* file that contains information about this transfer. The *MetaInfo* file is the same as in Table 3.1. The BS also creates a *Handshake* message and sends it to the requesting SM. The BS then starts sending pieces directly to the SM over a cellular link. A Save bit in the *Piece* message is always set to indicate that the SM should cache the received piece. The SM keeps a *BitField* to keep track of the received pieces. After sending all the pieces, the BS will send a *Done* message. When the SM receives the pieces and the *Done* message, it will send a message containing the *BitField* to the BS.

3.1.3 Distribute to SMs

As mentioned above, this case is only used with DISCS. In this case, a popular video file (for instance, a video file that was requested *n* times) is requested by a Non-SM (NSM) UE. The BS distributes the video file pieces over SMs and asks them to cache the pieces and forward them to the requesting UE (as the file is popular, and distributing it will be beneficial for the cluster). The download process (Figure 3.3) is as follows:

1) The UE sends a *Download Request* message to the BS.
2) The BS creates a *MetaInfo* file that describes the parameters for the download session of this video file (as in Table 3.1). The BS then sends a *Handshake* message (containing the *MetaInfo* file) to the requesting UE.
3) The BS then sends *Assistance Request* messages to the SMs of the cluster asking for their help to send the pieces to the requesting UE. There is a field in the message that is set to indicate that this is a "receive and forward" request, i.e., the SM is needed to receive the piece, cache it, and forward it to the requesting UE.

4) The SMs will send a *Response* message to indicate their availability for assistance, and to indicate the maximum number of outstanding assists the BS can send.

5) The BS then starts distributing the pieces to the SMs. Each *Piece* message has an index that identifies that piece.

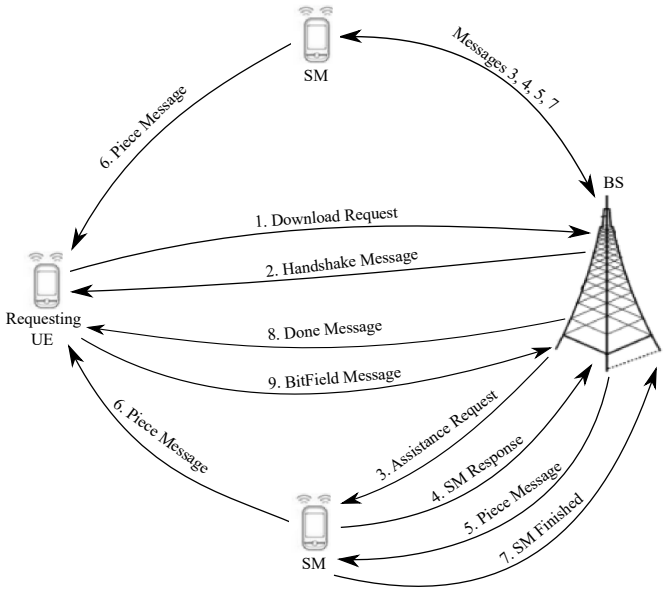

Figure 3.3. DTSMs case.

6) When an SM receives a piece, it will cache it, and send it to the requesting UE over a D2D link.

7) When an SM finishes sending piece(s), it will send an *SM_Finished* message to the BS, acknowledging the transmission of the piece(s).

8) When the BS receives *SM_Finished* for all the pieces from the SMs participating, it will send a *Done* message to the requesting UE.

9) When the requesting UE receives a *Done* message, it will send a *BitField* message to the BS indicating the pieces it has received.

This case helps speeding up accumulation of popular video files in the distributed cache of the cluster. It also allows for more parallelism and load balancing among SMs when sending video files from the distributed cache of the cluster. This should increase the utilization of the D2D channel and speed up the transmission, and consequently increase the average data rate.

In addition to the data the BS needs to keep in its database for CSVD (list of the clusters, list of the members and SMs of each cluster, and list of cached video files/segments), the BS keeps track of the number of times video files were requested recently.

3.1.4 Send to a UE

In this case, the requesting UE is not an SM, and the file is not cached. Hence, the BS will transmit the file directly to the requesting UE over a cellular link. This case is similar to STSM. However, in this case, the Save bit is always zero in the *Piece* message so that the piece will not be cached.

3.1.5 Segmented video download

We refer to the conventional download process in current cellular networks as Segmented Video Download (SVD), as video files are sent in pieces. In SVD, file caching and D2D communications are not used. Instead, video files are always sent as in STUE.

3.2 Analyzing the hit ratio of the proposed algorithms

In this section, we analyze the hit ratio of the proposed algorithms, i.e., the fraction of video segments that are found and satisfied from the cluster's cache. As mentioned before, the cell is divided into clusters. Only SMs in each cluster are used to cache video files. We assume that no video files are initially cached, and that video files are cached as

requested. A hit occurs if a requested video segment is delivered from the cluster cache (found in one of the SMs of that cluster). A miss occurs in the case the requested segment is not found in cluster's cache, and hence, it will be sent through the BS. We provide analytical models for video caching with both algorithms and develop closed-form expressions for the hit ratio of CSVD and DISCS. Furthermore, we run simulations using the DEVS model (presented in the next chapter) to evaluate the hit ratios.

In the following, we present the analytical models and the derivation of the expressions for the hit ratio of CSVD and DISCS. In Chapter 5, we present the results obtained from the analytical expressions as well as the simulation results from the DEVS model.

3.2.1 Hit ratio of CSVD

Consider a cell with N UEs in the cluster. Furthermore, consider that time is divided into slots, each with a length of T_s seconds. Each UE wants to request one video file, and each UE tries to generate the request during each slot with a probability, P_{req}, until it is generated. The UEs generate requests to download videos from a list that contains F video files. The popularity of videos is generated according to a Zipf distribution to simulate a variable popularity of files, as it has been established that this is a good model for video files popularity [83]. Using this distribution, some files are requested more often than others are. The Zipf exponent, β, controls the relative popularity of the files. According to the Zipf distribution, the relative popularity of the f^{th} file is,

$$P_f(\beta, F) = \frac{f^{-\beta}}{\sum_{i=1}^{F} i^{-\beta}}. \tag{3.1}$$

From now on, we will refer to the relative popularity of a file, i.e., the probability that a

UE selects file f as P_f. Without loss of generality, a file is assumed to be one piece. P_{SM} is the probability that a UE is an SM. This depends on the ratio of the central area of the cluster (see Figure 3.1) to the total cluster area. Consider that it takes R time slots for all the UEs to send their download requests.

The probability that a UE requests during slot r, P_r, follows a geometric distribution, as follows,

$$P_r = \left(P_{req}\right)\left(1 - P_{req}\right)^{r-1}. \tag{3.2}$$

As requesting in slot r and choosing file f are independent events, the probability that a UE requests during slot r, and selects file f, $P_{r,f}$, can be given as,

$$P_{r,f} = P_r \times P_f. \tag{3.3}$$

Similarly, the probability that a UE is an SM, that requests in slot r, and selects file f, $P_{SM,r,f}$, can be given by,

$$P_{SM,r,f} = P_{SM} \times P_r \times P_f. \tag{3.4}$$

When a UE sends a request in slot r to download video file f, the request will be a hit if an SM has requested the same file in one of the previous slots. As such, the probability a certain request that takes place in slot r is satisfied from the cluster's cache, $P_{Hit,r,CSVD}$, is,

$$P_{Hit,r,CSVD} = \sum_{f=1}^{F}\left(\Pr\left[\text{A UE requests file } f \text{ in slot } r\right] \times \Pr\left[f \text{ is cached by at least one SM}\right]\right). \tag{3.5}$$

Recalling that a file is either cached in the cluster's cache or not, $P_{Hit,r,CSVD}$ can be also give as,

$$P_{Hit,r,CSVD} = \sum_{f=1}^{F}\left(\Pr\left[\text{A UE requests file } f \text{ in slot } r\right] \times \left(1 - \Pr\left[f \text{ is not cached by an SM}\right]\right)\right). \tag{3.6}$$

Given that a file, f, is cached if it was requested in a previous slot by an SM, $P_{Hit,r,CSVD}$

can be also give as,

$$P_{Hit,r,CSVD} = \sum_{f=1}^{F}\Big(\Pr\big[\text{A UE requests file } f \text{ in slot } r\big] \times \Pr\big[\text{At least one SM requested } f \text{ in one of the prevous slots}\big]\Big)$$

$$= \sum_{f=1}^{F}\Big(\Pr\big[\text{A UE requests file } f \text{ in slot } r\big] \times \Big(1-\Pr\big[\text{No SM requested } f \text{ in one of the prevous slots}\big]\Big)\Big)$$

$$= \sum_{f=1}^{F}\Big(p_{r,f} \times \Big(1-P_{f \text{ not cached}|_{r},CSVD}\Big)\Big), \tag{3.7}$$

where $P_{f \text{ not cached}|_{r},CSVD}$ is the probability that a file, *f*, is not cached at time slot *r*, which can be give by,

$$P_{f \text{ not cached}|_{r},CSVD} = \Pr\big[\text{No SM requested file } f \text{ in one of the prevous slots}\big]$$

$$= \prod_{u=1}^{r-1}\binom{N-1}{0}\Big(P_{SM,u,f}^{0}\Big)\Big(1-P_{SM,u,f}\Big)^{N-1} = \prod_{u=1}^{r-1}\Big(1-P_{SM,u,f}\Big)^{N-1}. \tag{3.8}$$

Assuming that no files are cached in the beginning, only requests that take place at slot r = 2 or after can be satisfied from the cluster's cache. By substituting (3.8) in (3.7), and taking the summation over time slots 2 and after, the hit ratio for CSVD; the probability that a request is satisfied from the cluster's cache, $P_{hit,CSVD}$, can be given as,

$$P_{hit,CSVD} = \sum_{f=1}^{F}\sum_{r=2}^{R} P_{r,f} \times \left[1-\prod_{u=1}^{r-1}\Big(1-P_{SM,u,f}\Big)^{N-1}\right]. \tag{3.9}$$

3.2.2 Hit ratio of DISCS

For DISCS, consider *n* is the number of times a video file should be requested by UEs to be considered popular and to be distributed to SMs (DTSMs). In this case, a file *f* is delivered through the cluster's cache (SWA and DTSMs combined) if it was previously

requested by an SM, or if it was previously requested by NSMs *(n-1)* times. As such, with DISCS, a file *f* is not cached at slot *r*, if *f* was not requested by an SM in the previous time slots, and it was not requested by *n* or more NSMs in the previous time slots. Without loss of generality, consider *n = 2*. This means that if the video was requested once by an SM or was requested more than once by NSMs in the previous slots, it will be delivered from the cluster's cache (SWA). In this case, the probability that a file is not cached at time slot *r*, $P_{f \text{ not cached}|_r,DISCS}$, can be given by,

$$\begin{aligned} P_{f \text{ not cached}|_r,DISCS} &= \Pr[\text{file was not prevously requested by an SM}] \\ &\quad \times \Pr[\text{file was not prevously requested by NSMs more than once}] \\ &= \prod_{u=1}^{r-1}\left(1-P_{SM,u,f}\right)^{N-1} \times \Big[\prod_{u=1}^{r-1}\left(1-P_{NSM,u,f}\right)^{N-1} \\ &\quad + \sum_{u=1}^{r-1}\binom{N-1}{1}\left(P^{1}_{NSM,u,f}\right)\left(1-P_{NSM,u,f}\right)^{N-2} \times \prod_{\substack{j=1 \\ j\neq u}}^{r-1}\left(1-P_{NSM,j,f}\right)^{N-1}\Big] \\ &= \prod_{u=1}^{r-1}\left(1-P_{SM,u,f}\right)^{N-1} \times \Big[\prod_{u=1}^{r-1}\left(1-P_{NSM,u,f}\right)^{N-1} \\ &\quad + \sum_{u=1}^{r-1}(N-1)P_{NSM,u,f}\left(1-P_{NSM,u,f}\right)^{N-2} \times \prod_{\substack{j=1 \\ j\neq u}}^{r-1}\left(1-P_{NSM,j,f}\right)^{N-1}\Big]. \end{aligned} \tag{3.10}$$

As with CSVD, the probability a certain request that takes place in slot *r* is satisfied from the cluster's cache, $P_{Hit,r,DISCS}$, is given by,

$$\begin{aligned} P_{Hit,r,DISCS} &= \sum_{f=1}^{F}\Big(\Pr[\text{A UE requests file } f \text{ in slot } r] \times \Pr[f \text{ is cached by at least one SM}]\Big) \\ &= \sum_{f=1}^{F}\Big(\Pr[\text{A UE requests file } f \text{ in slot } r] \times \big(1-\Pr[f \text{ is not cached by an SM}]\big)\Big) \\ &= \sum_{f=1}^{F}\Big(\Pr[\text{A UE requests file } f \text{ in slot } r] \times \left(1-P_{f \text{ not cached}|_r,DISCS}\right)\Big). \end{aligned} \tag{3.11}$$

Assuming that no files are cached in the beginning, only requests that take place at slot r = 2 or after can be satisfied from the cluster's cache. By substituting (3.10) in (3.11), and taking the summation over time slots 2 and after, the hit ratio for DISCS; the probability that a request is satisfied from the cluster's cache, $P_{hit,DISCS}$, can be given as,

$$P_{Hit,DISCS} = \sum_{f=1}^{F}\sum_{r=2}^{R} P_{f,r} \times \Big[1-\Big(\prod_{u=1}^{r-1}\big(1-P_{SM,u,f}\big)^{N-1} \times \Big(\prod_{u=1}^{r-1}\big(1-P_{NSM,u,f}\big)^{N-1} + \sum_{u=1}^{r-1}(N-1)P_{NSM,u,f}\big(1-P_{NSM,u,f}\big)^{N-2} \times \prod_{\substack{j=1 \\ j\neq u}}^{r-1}\big(1-P_{NSM,j,f}\big)^{N-1}\Big)\Big)\Big]. \quad (3.12)$$

In Chapter 5, we present the results for the hit ratio of both algorithms that are obtained from the analytical expressions above as well as the simulation results.

3.3 Operation with UE mobility and implementation issues

Early in this chapter, we listed many scenarios where the proposed algorithms can be employed. Although users in such scenarios are usually static, some users (or all users sometimes) might be moving at pedestrian speed. For this reason, we discuss the operation of the proposed algorithms under user mobility in this section. As user mobility might reduce the performance improvement achieved by the proposed algorithms, we propose a minor update to the operation of the algorithms to be compatible with the case of user mobility. Afterwards, we discuss two important issues; the implementation of the algorithms in cellular networks, and the power consumption of SMs.

3.3.1 Operation of the algorithms with UE mobility

When UEs are moving, the BS needs to be aware of the location of the UEs. As such, moving UEs need to provide the BS with their updated location. This update can be sent periodically when the UE is continuously moving. As UEs are expected to be moving

around average pedestrian speed, these updates can be sent to the BS periodically on a scale of hundreds of milliseconds or even on a scale of seconds. If the UE is not continuously moving, such update can be sent only when the UE changes its location by a certain threshold (for instance, a couple of meters). The BS will need such information to update the clusters in the case any of the following changes occur:

- An SM moves out of the central area of the cluster
- An NSM moves into the central area of the cluster
- A UE moves from one cluster to another

A UE that is close to the cluster edge and moving towards the edge of the cluster will be marked as "transitioning". UEs that are marked as transitioning will be sent video segments only through the BS (not from the cluster's cache) until they transition to the new cluster.

The movement of UEs is expected to cause some degradation to the improvements achieved by the proposed algorithms. This is because under UE movement, SMs with cached content might now move out of the central area of the cluster and become NSMs.

To overcome this challenge, we propose a change to the operation of the proposed algorithms. Instead of setting the save bit to 1 only in the STSM and DTSMs cases, the save bit will always be set to 1. This means that every UE will be asked to cache received contents. This is because when UEs are moving, any UE could be an SM (after entering the central area) and hence it would be very beneficial for any UE that becomes an SM to have cached contents. This also increases the fairness of the algorithms, since in this case UEs that received contents from the cluster's cache could become SMs and provide others

with video contents later. In Chapter 5, we study the impact of UE movement on performance. Furthermore, we show how the improved operation overcomes this impact.

Although we focus in this thesis on the operation of the algorithms in a single cell, the proposed algorithms can be implemented over the area of multiple cells. To achieve this, the movement of UEs from one cell to another should be performed while maintaining the CSVD service. Considering that the target cell is employing the algorithms, the current serving BS gives the target BS the CSVD information during handover over the wired backhaul network. This includes information about the video currently being downloaded (if any) and information about delivered segments in the case the UE is currently requesting a video. This also includes information about currently cached video segments in the moving UE.

3.3.2 Implementing the algorithms in future cellular networks

In this work, we study the potential gains that can be achieved in cellular networks when the CSVD and DISCS algorithms are employed. The proposed algorithms focus on the RAN, which is the main bottleneck in cellular networks with limited frequency resources that are shared among a large number of users.

The algorithms can be implemented in cellular networks by employing an entity at the BS, namely the CSVD proxy. The CSVD proxy checks the requests from the clients to the content server. The CSVD proxy stops the requests when the contents are found in the cell, as per the proposed algorithms. This entity also performs the BS part of the employed algorithm (for example, steps 2, 3, 5, and 8 in Figure 3.3). In Chapter 7, we study and employ the proposed algorithms in the context of video streaming, and we

elaborate further on the implementation of the proposed algorithms in the context of video steaming over cellular networks.

The proposed CSVD and DISCS algorithms are more suitable for 5G cellular networks, as D2D communication is one of the technologies in 5G cellular networks. The algorithms can provide, in many scenarios, an alternative to other costly options such as cell densification and heterogeneous networks which involve the deployment of small cells that coexist with the macro BS [84]. Even in the case of heterogeneous networks, small cells are not available in all areas of the macro cell. As such, employing the proposed algorithms can be more beneficial in such areas and would result in significant performance improvements. It is worth mentioning that in the case the algorithms are employed in heterogeneous networks, they will be implemented in the macro BS as it covers a wider geographical distance.

Obviously, the proposed algorithms introduce extra processing to the BS in LTE-A networks. This extra processing is needed to implement the different aspects and functionalities of the proposed algorithms such as looking up requested segments and assignment of SMs to requesting UEs. However, BSs nowadays are equipped with high processing capabilities to perform complex tasks such as baseband processing and control of radio resources [85]. Such BSs are capable of implementing the functionalities required by the proposed algorithms. For instance, Orthogonal Frequency-Division Multiple Access (OFDMA)-based resource allocation in current LTE-A networks, which involves allocation of frequency resources to active users, is performed every 1 ms. Allocation of SMs to requesting users is a doable task that can be performed on a larger time scale. In Chapter 9, we provide a more detailed analysis on the processing time

needed for such task. Furthermore, different techniques can be used to manage the processing overhead required by the proposed algorithms. For instance, a limit can be imposed on the number of assistance requests transmitted in certain period.

The proposed algorithms introduce some transmission overhead as well. This is because implementing the algorithms require transmission of extra messages (e.g., *Assistance Request* message). In Chapter 5, we provide analysis for the overhead of the proposed algorithms. Results show that significant performance improvements can be achieved by the proposed algorithms, and the overhead introduced by them is negligible compared to the performance improvement achieved.

Regarding the transmission of data over different interfaces, this can be achieved using new technologies in the 5G cellular networks such as multi-RATs, dual connectivity, and flow splitting/aggregation [18]. With flow splitting/aggregation dual connectivity, the user data can be transmitted simultaneously over different air interfaces. To achieve this, the user plane is split over multiple interfaces. Separate PHY, Medium Access Control (MAC), and Radio Link Control (RLC) layers are used for transmission over each interface. Furthermore, transmissions over different interfaces are aggregated using a common Packet Data Convergence Protocol (PDCP) layer [18] at the receiver.

With respect to SMs transmission parallelism, different SMs in the cluster can transmit simultaneously if their concurrent transmissions take place over different subchannels. In our work, we assume that multiple D2D subchannels are available and these are allocated to the SMs by the BS.

3.3.3 Power consumption of the SMs

Power consumption is an issue with D2D content sharing in general (not just with the proposed algorithms). This is due to the fact that receivers get contents while helpers have to consume energy. The power consumption of the SMs should not be a major problem for the applicability of the proposed algorithms, for the following reasons:

1) The architecture employed by the proposed algorithms is designed to shorten the distance between the SMs and the requesting UEs. Due to clustering and selection of SMs in the central area of the clusters, SMs are relatively close to requesting UEs in the cluster. Because of this proximity between SMs and requesting UEs, SMs will not need to transmit with high transmission power, which reduces the power consumption.
2) Work in the literature on D2D communication usually assumes helpers are willing to participate. However, helpers in D2D communication may not comply with the requested assistance, especially considering power consumption. For this reason, providing incentive mechanisms for D2D communication is a topic that has been investigated by researchers. There are some existing incentive mechanisms to motivate users' involvement (especially helpers) in D2D communication. For instance, some of these mechanisms assume that helpers are rewarded by the network operator, for example with money, better data plan, or other types of rewards. As previously mentioned, we do not consider incentive mechanisms here, as this is a different research area that is out of the scope of this paper. The interested reader is referred to [34], [35] for further information on this topic.
3) With the proposed algorithms, SMs in the cluster could request video contents

and get help from other SMs in the cluster, which can be an incentive for SMs to get involved. Furthermore, the improved operation increases the possibility of a requesting UE becoming a helper/provider in the future and sending contents to another UE that was an SM.

4) Smart phones nowadays are equipped with batteries that last for a day, and can handle transmission of some video segments without losing much of their battery power. For instance, a simple test we performed with an iPhone 6s has shown that uploading a 1 minute video twice, once over a cellular connection and once over a Wi-Fi connection consumes less than 1% of the battery power.
5) Different policies can be employed with the proposed algorithms to avoid over usage of the SMs power. For instance, an SM with a low power level can signal its unavailability via the *Response message*.

3.4 Summary

In this chapter, two algorithms are proposed for BS-assisted progressive caching of video segments in UEs and P2P transmission of video contents among UEs using D2D communication. The algorithms are called CSVD and DISCS. Furthermore, analytical models for video caching with both algorithms are provided and closed-from expressions are derived for the efficiency of both algorithms in terms of the hit ratio. Finally, the operation of the algorithms under UE mobility and various implementation issues are discussed.

Chapter 4

DEVS-based modeling of the LTE-A network

In this chapter, we start by discussing the developed DEVS model for an LTE-A network that employs the proposed algorithms. Furthermore, we show how the model was implemented with the CD++ toolkit. Thereafter, we present some of the tests performed for model validation.

4.1 DEVS model

Figure 4.1 shows the structure of the LTE-A network DEVS model. The top level is called the *Cell* coupled model, that contains the *BS*, Transmission *Medium*, and many *UE* coupled models. It also contains the *Cell Manager* and *Log Manager* atomic models.

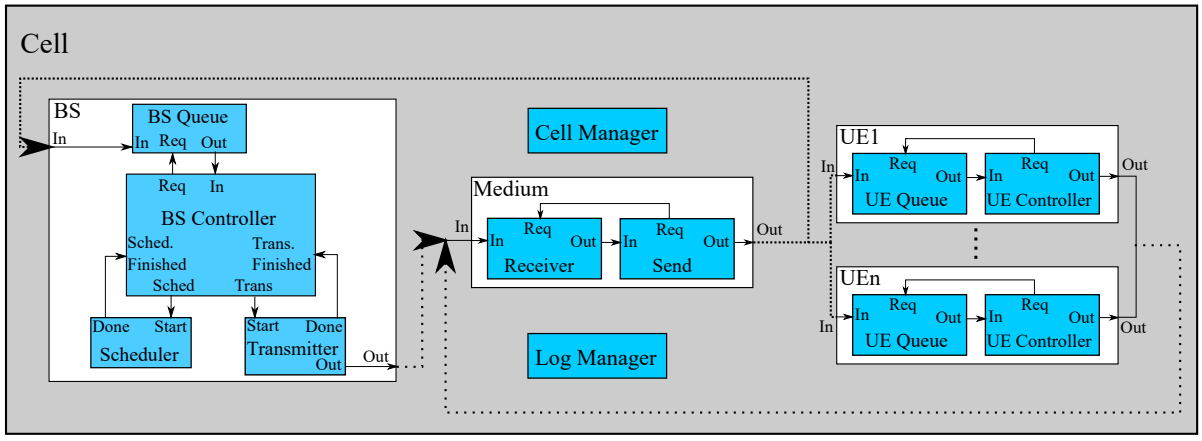

Figure 4.1. DEVS model of the cellular network.

The *BS* coupled model corresponds to the BS in the cell. The BS manages all the data transmission in the cell, the sessions with all the UEs, and allocates the radio frequency

resources to the UEs. It contains four atomic models: *BS Queue*, *BS Controller*, *Scheduler*, and *Transmitter*. The *BS Queue* atomic model is where received messages are buffered. The BS queue also checks the destination address of arriving messages. It buffers a message with destination address that matches that of the BS; otherwise, it ignores the message. The *BS Controller* processes received messages and operates as per the employed algorithm (e.g., CSVD). Every Transmission Time Interval (TTI), which is 1 ms, the BS processes received messages and asks the *Scheduler* to allocate frequency resources in the next TTI to active UEs in the cell. Every TTI, the *BS Controller* also asks the *Transmitter* to send data that was scheduled for transmission during this TTI.

The *UE* coupled models represent the UEs in the cell. A *UE* coupled model contains two atomic models: *UE Queue*, and *UE Controller*. The UE Queue is where received messages are buffered. The *UE Controller* is where the UE part of the algorithm is implemented.

The *Medium* model receives a message sent from the BS or any UE and broadcasts it to the other receivers (BS/UEs) in the cell. Receivers recognize their intended messages by the destination address filed in the message. As mentioned above, this is done at the *BS Queue* and *UE Queue* models.

The *Log Manager* logs simulation events and record statistics during the simulations. The *Cell Manager* atomic model initializes and updates the parameters of the cellular DLs and uplinks (ULs) between the BS and the UEs, as well as the D2D links between the UEs (e.g., path loss and received signal power). Path loss and shadowing are considered here.

The urban macro propagation model [86] was used for cellular links with a DL operating carrier frequency of 900 MHz, and a transmission bandwidth of 10 MHz. According to [86], the propagation model, L, is given by,

$$L = 40\times\left(1-4\times10^{-3}\times Dhb\right)\times log_{10}\left(d\right) - 18\, log_{10}\left(Dhb\right) + 21\, log_{10}\left(f\right) + 80dB, \quad (4.1)$$

where d is the BS-UE separation in kilometers, f is the carrier frequency in MHz, and Dhb is the BS antenna height in meters, measured from the average rooftop level. The path loss, PL, then can be calculated as,

$$PL = L+LogF, \quad (4.2)$$

where $LogF$ is a lognormally distributed shadowing with standard deviation of 10 dB. The received signal then can be calculated as,

$$P_{RX} = P_{TX} - MAX(PL - G_{TX} - G_{RX}, MCL), \quad (4.3)$$

where P_{RX} is the received signal power, P_{TX} is the transmitted signal power, G_{TX} is the transmitter antenna gain, G_{RX} is the receiver antenna gain, and MCL is the minimum coupling loss.

Considering Additive White Gaussian Noise (AWGN), the link data rate, R, can be calculated as,

$$R = B\times\log_2(1+\frac{P_{RX}}{P_n}), \quad (4.4)$$

where P_n is the noise power and B is the transmission bandwidth.

For D2D transmission, we used the D2D channel model at 24 GHz, defined in [87], and the data rate is calculated considering AWGN, as in (4.4) above.

As previously mentioned, each atomic component represents the behavior of a part of the system. The *BS Controller*, for example, checks its queue for messages from UEs. The BS processes received messages and updates the state of the corresponding download session. For instance, if the received message is a new download request, the BS will create a new *Session* object to that node. If the received message is an *SM_Finished* message, the BS will update the statistics of that session (number of transmitted pieces, transmitted bits, etc.). We will provide more details on the behavior of this model in the next section and show how it is implemented in CD++.

4.2 Implementation of the DEVS-based model

We used the CD++ toolkit to implement our LTE-A network DEVS model, which is an open-source simulation software written in C++ and implements the DEVS abstract simulation technique. The simulation engine tool of CD++ is built as a class hierarchy. With CD++, atomic models are developed using C++ programming language and can be incorporated into the class hierarchy. Figure 4.2 depicts a simplified Unified Modeling Language (UML) class diagram that shows the main classes of our model. As can be seen from Figure 4.2, the *BS Controller* atomic model is implemented with the *BS* class. The attributes of this class are used to contain the specifications of a BS, such as the location, maximum transmission power, antenna gain, etc. The class also has functions necessary to implement the functionalities performed by the *BS Controller*, as per the algorithms.

Similarly, the *UE* class is used to represent the *UE Controller* atomic model. It has many

attributes to contain the parameters of the UE (e.g., location, transmission power, antenna gain, etc.) and many functions to implement the functionalities of the UE.

Figure 4.2 shows that in addition to the atomic models above, many other passive classes where developed to model other components of the system such as classes to model the cellular DLs and ULs, D2D links, download sessions the BS has with UEs, cell clusters, exchanged message, etc. The *Link* base class is used to represent transmission links.

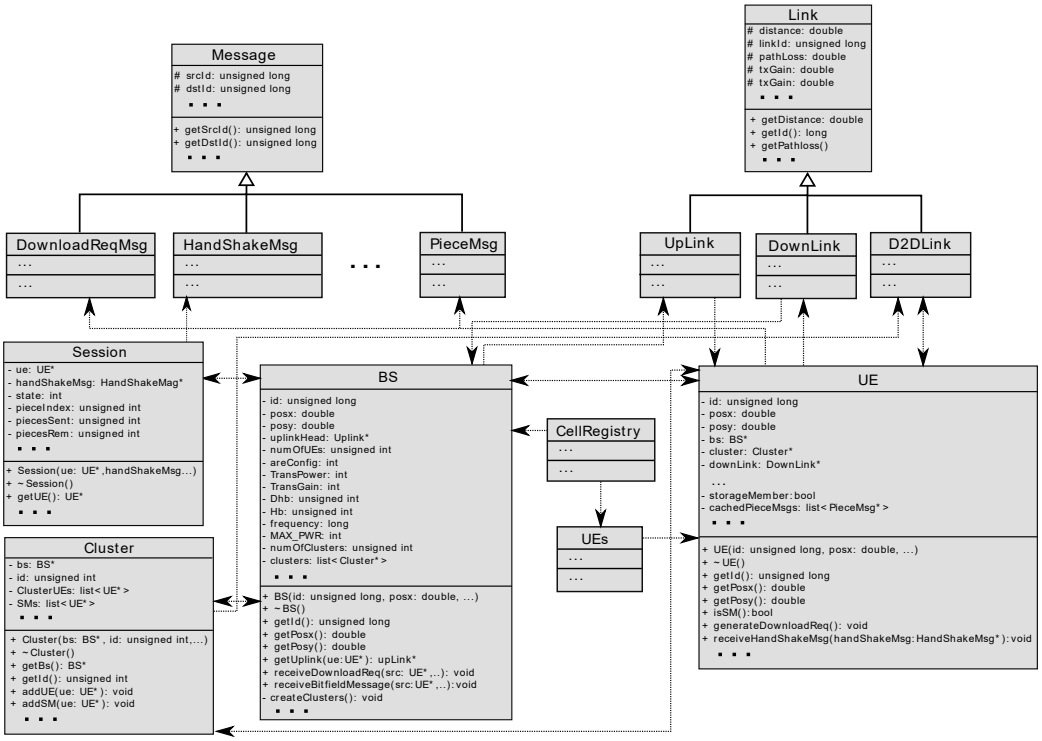

Figure 4.2. Simplified UML diagram of the DEVS model.

It has many attributes to represent the transmission link parameters such as distance, path loss, received power, etc. The derived classes *DownLink*, *UpLink*, and *D2DLink*, are used to implement the DL, UL, and D2D links, respectively. These derived classes contain certain parameters and functions to compute the metrics (e.g., path loss) of the corresponding link type. Due to lack of space in the figure, many other classes are not

shown such as classes to implement the *Queue*, *Transmitter*, and *Scheduler* atomic models.

```
BS::BS(const string &name) : Atomic(name),
  in(addInputPort("in")), req(addOutputPort("req")),
  schedFinished(addInputPort( "schedFinished" )),
  sched(addOutputPort( "sched" )),
  transFinished(addInputPort( "transFinished" )),
  trans(addOutputPort( "trans" )){
  ...   }

Model &BS::externalFunction(ExternalMessage &msg){
  receivedMsg = msg.valueO();
  int msgType;
  msgType = receivedMsg->getMsgType();
  const Time msgTime = msg.time();
  if (state == CHECK_QUEUE){
    switch(msgType){
     case MSG_DOWNLOAD:
      dlMsg = (DownloadMsg*)receivedMsg;
      state = RCV_MSG;
      keepHoldInTime = receiveDownloadReq
                           (dlMsg, msgTime);
      this->getSessionPtr(receivedMsg->getSrcID())
      ->state = RCV_DOWNLINK_REQ; break;
 ... }
  }
}

Model &BS::internalFunction(InternalMessage &msg ){
  switch(state){
    case INITIAL:
      state = CHECK_QUEUE;
      this->bs->holdInActive(Time::Zero); break;
    case CHECK_QUEUE:
      this->bs->passivateBS(); break;
    case RCV_MSG:
      state = CHECK_QUEUE;
      bs->holdInActive(keepHoldInTime); break;
    ...
  }
}

Model &BS::outputFunction(InternalMessage &msg){
  switch(state){
    CHECK_QUEUE:  //request the next message
      bs->sendReq(msg.time(), 1, NULL); break;
    RCV_MSG:
      if(receivedMsg != NULL){//delete msg from queue
         bs->sendReq(msg.time(), 2, NULL);} break;
  ...
  }
}
```

Figure 4.3. Code snippet from the *BS Controller* atomic model in CD++.

A sample code snippet is shown in Figure 4.3, which includes small parts of the implementation of the *BS controller* atomic model (class *BS*). The model has five states:

Inital, Check_Queue, Rcv_Msg, Schedule, and Send. At the beginning of each iteration, the model is in the *Initial* state as can be seen in the internal function. Then it goes to the *Check_Queue* state, during which, the BS sends requests (as shown in the output function) to the queue asking for the messages from UEs. The queue will send the next message in line. When the model receives a message (see the external function), it goes to *Rcv_Msg* state. During this state, the BS processes the message, updates the state of the corresponding download session, and then sends another request to the Queue. The external function shows, as an example, the receipt of *MSG_DOWNLOAD*, which is the message sent by a UE to download video segments. When no more messages are available, the queue sends *EMPTY_QUEUE* message to the *BS controller*. When the *BS controller* receives *EMPTY_QUEUE* message, it goes to the *Schedule* state. In this state, the BS sends a message to the *Scheduler* to schedule the data to be sent during the next TTI. When the scheduler finishes scheduling, it sends a *SCHED_FINISHED* message to the *BS Controller* to indicate that scheduling is finished.

When the *BS Controller* receives the *SCHED_FINISHED* message, it goes to the *Send* state. In this state, the model sends a message to the *Transmitter* model to start transmission. When the transmission is done, the *Transmitter* model sends a *TRANS_FINISHED* message to the BS controller to indicate that transmission is finished. Upon receipt of this message, the BS goes back to the *Check_Queue* state.

With CD++, coupled models can be created using a language built in the simulation engine. Figure 4.4 shows a code snippet from the DEVS coupled model file. As can be seen from Figure 4.4, the coupled model file is used to create the coupled model and maintain its hierarchical structure. For each coupled model, the components are listed and

the links between these components are defined. For instance, the components for the top coupled model (*Cell*) are listed as: *logManager*, *BS*, *UE*, etc. Some of the defined links are shown. For instance, an out from the *Medium* coupled model is used as an input to the *BS* coupled model (as in Figure 4.1). The parameters for the atomic models are also listed in the file. For instance, the code shows some parameters for the *BS Controller* atomic model.

```
[top]
components : logManager@LogManager  BS  Medium  UE1 ... UE100
cellManager @CellManager
...
Link : out@BS in@Medium
Link : out@Medium  in@BS
Link : out@Medium  in@UE1
Link : out@UE1 in@Medium
...

[BS]
components  :  BS1Queue@BSQueue  BS1Controller@BSController  scheduler@Scheduler
transmitter@Transmitter
[BS1Controller]
numberOfUEs : 100
BSId : 1
posX : 0
posY : 0
...

[UE1]
components : ue1Queue@UEQueue ue1Controller@UEController
...
```

Figure 4.4. Code snippet from the DEVS coupled model file.

4.3 Verification and validation of the model

Simulation models are increasingly adopted to study and understand the behaviour of real life systems, and to assist in decision-making processes. A model is usually developed for a certain purpose and used to study some aspects of the system. Model Verification and Validation (V&V) are concerned with the correctness of the developed model with respect to the purpose of that model [88]. Model verification is testing the computer program of the computerized model to ensure that it is implemented correctly. Model

validation is ensuring that the developed model has a satisfactory range of accuracy within the intended application of that model [88].

Here, we discuss the approaches used for model V&V and present some of the validation tests that we carried out to ensure the correctness of our DEVS model. Regarding model verification, we performed thorough debugging, structured walk-through, and step-by-step analysis of the model in a bottom-up fashion. The modular and hierarchical nature of DEVS is a very useful property for implementing this bottom-up approach. Our network model is built using different submodels where each one implements a certain component of the wireless network such as the BS or the UE. Each one of these submodels is built using an atomic DEVS model or a smaller coupled model. As such, each atomic model is verified via thoroughly debugging, structured walk-through, and step-by-step testing. After verification of all the atomic models in a coupled model, the coupled model itself is tested. Similarly, after verification of all the coupled submodels in a coupled model, the integration of the coupled submodels is tested. This thorough testing is performed for every atomic and coupled submodel in a bottom-up fashion until the top-level coupled model. With respect to the employed DEVS simulation algorithms, they are formally validated and shown to implement the DEVS models as intended [20], [21], which is another advantage for using the DEVS methodology.

Regarding model validation we used the popular three-step validation approach based on the work of Naylor and Finger [89]. Such approach consists of the following three steps:

- Face validation
- Validation of model assumptions

- Validation of the input/output transformations

To carry out the steps above, we executed various subjective and objective evaluations. In the following, we present how each one of the steps above was implemented.

4.3.1 Face validation

With face validation, the model is checked by a domain expert to see if it is reasonable. This can be done before implementation to check model assumptions and after implementation to assess the behaviour of the model.

Our DEVS model was developed in collaboration with experts from Ericsson Canada. In the early stages of model implementation, domain experts from Ericsson Canada evaluated our conceptual model to confirm that the assumptions and simplifications adopted are reasonable, and to ensure the correctness of the developed model with respect to its purpose. We are mainly interested in modeling resource allocation in the RAN, which is the main bottleneck in cellular networks with limited frequency resources that are shared among large number of users. Another purpose of the model is to study the potential performance improvements achieved by employing the proposed algorithms for progressive caching and D2D communication of video contents, and compare the performance achieved by the two algorithms. Furthermore, we want to use the model to study the impact of different parameters on the performance of the proposed algorithms.

Further discussions with the experts took place during and after model implementation to discus the behaviour of the model and validate the obtained simulation results.

4.3.2 Validation of model assumptions

Model assumptions are categorized into structural and data assumptions [90]. Structural assumptions are related to the operation of the system, while data assumptions are related to behavioural assumptions that are based on data collection.

To ensure validity of the structural assumptions of our model, we adopted such assumptions and parameters from trusted sources (books and standards) on the LTE-A system. Here we present some examples of structural assumptions used in our model regarding resource allocation that were adopted from the MAC scheduler interface specifications in the LTE-A standard,

- The LTE scheduler is located at the BS
- OFDMA is used for cellular resource allocation
- The subframe module (implemented in the BS controller) triggers MAC scheduling at the beginning of each TTI which equals to 1 ms
- Time slot is the smallest unit of allocation in time and it equals to 0.5 ms
- scheduling decisions are returned to the subframe module consists of resource mapping between UEs and RBs

We also used some data assumptions in our model. These assumptions are made by other researchers and we adopted them. We made sure that such assumptions are adopted from trusted sources and based on large data sets and correct statistical analysis. Example of such assumptions are the models used for the popularity of videos and the size of requested video files. In addition to ensuring that such assumptions are either common or adopted from trusted sources, they were also approved by the experts from Ericsson

Canada, as mentioned above.

4.3.3 Validation of the input/output transformations

In this step, the model is treated as input/output transformations, i.e., it takes input values and generates output values. There are many tests that can be used to evaluate the input/output transformations of a computerized model and ensure its operational validity [88]. Operational validation is performed to ensure that the model's output behavior has the accuracy required for the model's intended purpose within the scope of its intended applicability [88]. We have performed many of these tests on our model. In the following, we present some of the performed tests.

1) Degenerate test

In this test, the degeneracy of the model's behaviour is evaluated using a set of input parameters. For instance, such test is performed on crowd models to check if pedestrian's speed decreases by increasing the density of people in the area.

With our DEVS model, we checked the produced average data rates per user in the cell versus the number of UEs in the cell under fixed cellular channel bandwidth (e.g., 10 MHz). In this case, the input is the number of UEs in the cell and the output is the average data rate. As expected, the average data rate per user decreases by increasing the number of UEs in the cell. This is because the available frequency resources are shared by higher number of users. This can be clearly seen in the SVD data rate results in Figure 5.6.

2) Extreme condition test

In this test, the model is tested under extreme factors or input values to ensure that the output is plausible. We tested our model under very high traffic loads to check if the

aggregate data rate keeps increasing under fixed cellular channel bandwidth (e.g., 10 MHz). As expected, under high traffic load and shared cellular channel with limited bandwidth, the aggregate data rate does not increase by increasing the number of active users, because these users share a cellular channel with limited bandwidth. This can be clearly seen in the SVD aggregate data rate results in Figure 5.5.

3) Fixed values test

In this test, the input variables and the system parameters are set to constant values. The computerized model is executed, and the output is compared to expected values. With respect to our previous example on pedestrian models, this can be performed by setting the speed of a pedestrian in the model to a constant value and setting up an experiment where the pedestrian walks a fixed distance. After executing the simulation model, the time needed for the pedestrian to walk that distance can be compared to the expected value. We used this test in combination with the next technique (compare to another model). The results of the test are shown in the discussion of that technique below.

4) Compare to another model (aggregate and average data rates)

Here, we use a similar approach to the one employed in [91] for model validation. With this approach, performance results obtained from the simulation model are compared to theoretical results in certain scenarios. The set of scenarios is selected beforehand. Furthermore, some simplifying assumptions are also considered to eliminate randomness and obtain deterministic theoretical results. Simulations are then executed with the model under the same set of scenarios and considered assumptions. Then, simulation results are compared with theoretical results to ensure operational validity of the simulation model.

As performance metrics, we consider the cell's aggregate data rate as well as the average data rate per user. As the theoretical transmission model, we use the urban macro propagation model [86] for the cellular channel at DL operating carrier frequency of 900 MHz, channel bandwidth of 10 MHz, and distance of 300 meters. Simulations were executed with our DEVS model with the same parameters. 300 UEs are considered in the simulations. All UEs have the same distance form the BS (300 meters) to match the setup used for the theoretical results. All UEs generate requests to download files at the same time (beginning of the simulation) and each UE generates 1 request. All the files have the same size of 50 Mbyte.

According to the reference model, the transmission rate obtained with 10MHz channel and 300 meters distance is 162 Mbps. According to the simulations, the aggregate transmission rate is 161.98 Mbps. As one can see, the results are very close, which validates the operation of our DEVS model (scheduling, transmission, etc.).

In addition to the aggregate data rate, we also compare the results for the average data rate per user. In the simulation scenario above, all the UEs in the cell request file download at the same time and all requested files have the same size. Furthermore, the BS transmit to all users with the same transmission power and all users have the same distance from the BS (300 UEs). As the BS employs round robin scheduling, all the UEs should have similar values for the average data rates. Furthermore, the aggregate rate divided by the number of UEs in the cell should give a close value to the average data rate per user value obtained from the simulations. The value obtained from the aggregate transmission rate divided by the number of UEs equals to 0.53993 Mbps. The average data rate value obtained from the simulations equals to 0.53398 Mbps. Again, the results

are very close, which provides further validation for the operation of our DEVS model (scheduling, transmission, etc.) as it generates the expected results.

5) Compare to another model (hit ratios)

In the previous chapter, we analyze the hit ratio of the proposed algorithms, i.e., the fraction of video segments that are found and satisfied from the cluster's cache. A hit occurs if a requested video segment is delivered from the cluster's cache (cached in one of the SMs of that cluster). A miss occurs in the case the requested segment is not found in cluster's cache, and hence it will be sent through the BS. We provide analytical models for video caching with both algorithms and develop closed-form expressions for the hit ratio of CSVD and DISCS. We run simulations using the DEVS model to evaluate the hit ratios of the proposed algorithms. In the next chapter, we present the results obtained from the analytical expressions as well as the simulation results from the DEVS model. As we will see in the next chapter, the simulation results are very close to the analytical results, which validates our developed LTE-A network DEVS model.

4.4 Summary

In this chapter, we present our DEVS model for an LTE-A network that employs CSVD and DISCS. Then, we discuss the implementation of our model with the CD++ toolkit. Finally, we present the approach and some of the tests performed for model V&V.

Chapter 5

Performance evaluation of CSVD and DISCS

5.1 Hit ratio results

In this section, we present the results for the hit ratio of the CSVD and DISCS algorithms obtained from both the analytical model and the simulations performed using the DEVS model.

In each simulation run, UEs are placed throughout the cell with a uniform distribution. The central area of each cluster forms 1/4 of the total area of the cluster. Hence, roughly, one fourth of the UEs in each cluster will be SMs. At the beginning of each time slot, which is 10 seconds, each UE tries to send a request with a probability, $P_{req} = 0.2$, or wait till the next time slot. Simulation ends when all UEs send their requests and download the video files. Each UE will request one video file. Simulations were performed with various numbers of UEs in the cluster. A list of 500 files was used in the simulations. 100 simulation runs were used. In addition to the mean values for the hit ratios obtained from the simulations, we show the Margin of Error (MoE) values with 95% confidence interval.

Figures 5.1 and 5.2 show the analytical and simulation results for the hit ratio of CSVD and DISCS, respectively. As can be seen from the figures, the simulation results are very close to the analytical results, which means that the analytical model accurately represents the transmission scenario used. This also verifies and validates our developed

LTE-A network DEVS model.

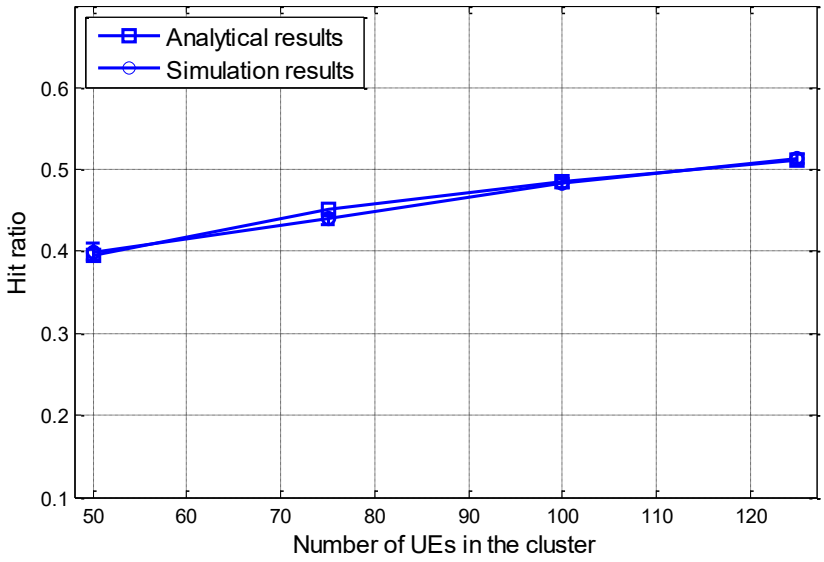

Figure 5.1. Hit ratio of CSVD versus number of UEs in the cluster. Zipf exponent = 1.5.

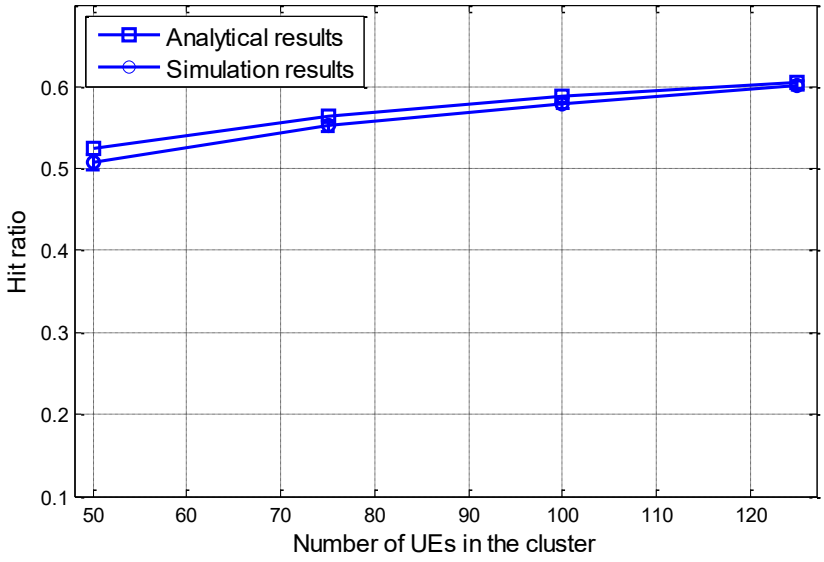

Figure 5.2. Hit ratio of DISCS versus number of UEs in the cluster. Zipf exponent = 1.5.

One can also see that for both CSVD and DISCS, the hit ratio increases by increasing the number of UEs in the cluster. This is because, with both algorithms, increasing the number of UEs increases the number of requests for video files and the number of SMs in each cluster. This increases the number of cached files in a cluster and the number of requests

that would be satisfied from the cluster's cache, which consequently increases the hit ratio.

Figure 5.3 shows the hit ratio (analytical results) for both algorithms versus the number of UEs in the cluster. As can be seen, the hit ratio for DISCS is always higher than that for CSVD. This is because in DISCS, video files are cached with the DTSMs case, in addition to the STSM case. As such, more video files will accumulate in the cluster's cache faster, which increases the number of video files satisfied from the cluster's cache, and increases the hit ratio.

As mentioned above, the Zipf distribution is used to simulate variable popularity for a group of video files. The Zipf distribution has one parameter, namely the Zipf exponent. This exponent controls the relative popularity of files. When the Zipf exponent increases, the popularity of some files in the group increases, which means that these files will have higher probability of being requested. This also increases the number of relatively popular files.

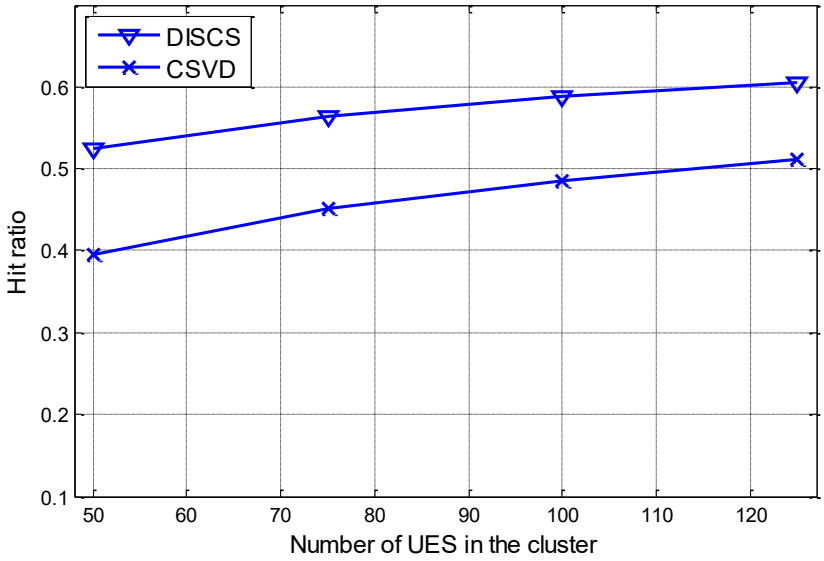

Figure 5.3. Hit ratio of CSVD and DISCS (Analytical results) versus number of UEs in the cluster. Zipf exponent = 1.5.

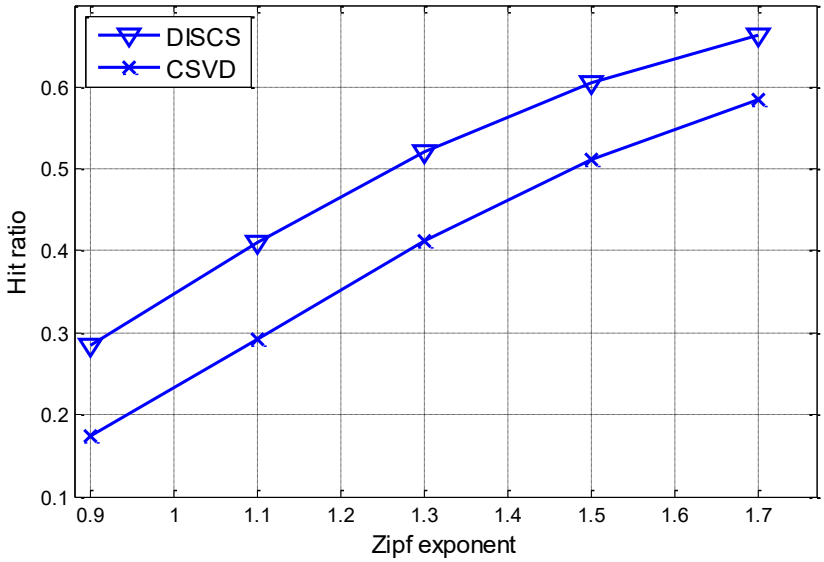

Figure 5.4. Hit ratio of CSVD and DISCS (Analytical results) versus Zipf exponent. Number of UEs in the cluster = 125.

Figure 5.4 shows the hit ratio (analytical results) for both algorithms versus the Zipf exponent. As can be noticed, the hit ratio increases by increasing the zipf exponent. As the popularity of some files increases, the utilization of the distributed cache increases, because more files will be cached and consequently delivered later from the distributed cache rather than through the BS. This increases the percentage of video files satisfied from the cluster's cache, i.e., increases the hit ratio.

5.2 Evaluation of the data rates of the proposed algorithms

System-level simulations were performed to evaluate the performance of CSVD and DISCS, and compare them to SVD in terms of the DL cell's aggregate data rate and average data rate per user. Recall that SVD is the conventional download process in current cellular networks (without caching nor D2D communication). We use more realistic transmission scenarios than the ones in the previous section to evaluate the data rates of the proposed algorithms in both the case of stationary UEs and the case of UE

mobility.

5.2.1 Simulation scenarios

In the Simulations, we consider a single LTE-A cell. The urban macro propagation model [86] was used for cellular links with a DL operating carrier frequency of 900 MHz, and a transmission bandwidth of 10 MHz. Table 5.1 shows the simulation parameters we used.

Table 5.1. Simulation setup.

Parameter	Value
Cellular channel BW (MHz)	10
Cell range (m)	500
BS antenna gain (dB)	12
BS transmission power (dBm)	43
UE antenna gain (dB)	0
UE transmission power (dBm)	21
Noise spectral density (dBm)	-174
Antenna height (m)	15
Transmission model	UTRA-FDD
Carrier frequency (MHz)	900
File size range (MB)	1-100
Area configuration	Urban
Piece size (KB)	512
Number of files	500
D2D channel BW (MHz)	60
D2D carrier frequency (GHz)	24
D2D transmitter TX Power (dBm)	23
D2D large-scale fading std deviation (dB)	4.3
UE receiver noise figure (dB)	9
D2D TX/RX height from ground (m)	1.5

In the beginning of each iteration of the simulations, the UEs are uniformly distributed throughout the cell. The cell is divided into 9 clusters. According to their location in the cell, UEs are assigned to clusters as shown in Figure 3.1. Furthermore, the UEs in the central area of each cluster are marked as SMs. The central area of each cluster forms 1/4 of the total area of the cluster. Hence, roughly, one fourth of the UEs in each cluster will

be SMs. Each iteration in the simulations is divided to two phases; a transient phase, followed by a steady-state phase. At the beginning of the transient phase, there are no files cached in the clusters. As UEs download videos during the transient phase, video segments will accumulate in the distributed cache of each cluster. At the beginning of the steady-state phase, there will be many pieces in the distributed caches of the clusters that were cached during the transient phase. During each phase, each UE sends 2 download requests in total (i.e., each UE downloads 2 video files). A UE sends one request at a time, and after downloading the video file, it generates another request. The arrival of requests is generated according a Poisson arrival process. At the end of each phase, we calculate the cell's aggregate data rate and the mean of the average data rate per user. The mean of the cell's aggregate data rate and the mean of the average data rate from all the iterations are calculated at the end of the simulations. The results show the mean values based on 50 simulation runs along with the MoE for 95% confidence interval.

The UEs generate requests to download video files from a list. The popularity of videos is generated according to a Zipf distribution to simulate a variable popularity of video files. It has been shown in [83] that this is a good model for relative popularity of videos. The distribution of the size of YouTube video files have been analyzed in some studies [92], [93], as it is the most popular video sharing service. These studies show that the lognormal distribution can be used for modeling video file size characteristics. Here, the size of the video files is generated according to a lognormal distribution as in [93]. Unless stated otherwise, the number of UEs is 500, the Zipf exponent is 1.5, and the number of requests made by a UE during each phase is 2. In Section 5.2.2, we consider the case of stationary UEs. In Section 5.2.3, we consider the case of moving UEs.

5.2.2 Simulation results

Figure 5.5 shows the Cell's aggregate data rate versus the number of UEs in the cell, for the SVD, CSVD, and DISCS algorithms, respectively, in the steady-state phase. Recall that SVD is the conventional download process in current cellular networks (without caching nor D2D communication). Up to one copy of each piece of a file is cached in a cluster in the case of CSVD and DISCS. As Figure 5.5 shows, CSVD and DISCS provide significant improvement over the SVD. The maximum aggregate rate achieved using the SVD is around 130 Mbps, while with the CSVD and DISCS, aggregate rates of 490 Mbps and 693 Mbps can be achieved, respectively, at 700 UEs. This significant improvement on the aggregate data rate is due to having more resources, i.e., the D2D channel with large bandwidth (60 MHz) available in each cluster, and used for D2D communication.

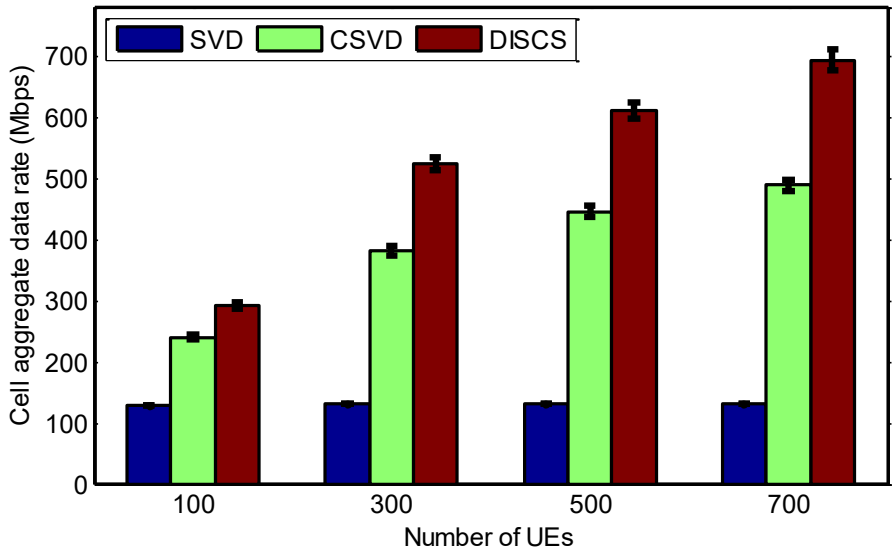

Figure 5.5. Cell's aggregate data rate vs. number of UEs in the cell (steady-state phase). 2 requests per user and Zipf exponent = 1.5.

Furthermore, DISCS achieves significant improvement over CSVD. This is because DISCS speeds up video caching and achieves a better hit ratio as discussed in the previous section, which improves the utilization of D2D channel. This is also because in

CSVD, when a video file is cached in a cluster, it is always cached in one SM, while in DISCS, a cached file is distributed over many SMs in the cluster in the case of DTSMs. As such, when video files are transmitted from the distributed cache, multiple SMs will be sending pieces in parallel to the requesting UE in the case of DISCS. As such, the D2D channel will be further utilized and the aggregate data rate will increase.

Figure 5.5 also shows that with CSVD and DISCS, the aggregate data rate increases with increasing the number of UEs in the network. Increasing the number of UEs increases the number of requests for video files and the number of SMs in each cluster. This increases the number of cached files in a cluster and the number of requests that would be satisfied from the cluster's cache. Hence, the D2D channel will be further utilized and the aggregate data rate will increase. With SVD, the aggregate data rate does not increase with the number of UEs in the cell. In SVD, each cell has fixed cellular resources (a 10 MHz channel is used here) and as the number of UEs increases, the utilization of the cellular channel will increase, until it is fully utilized. As such, we can say from Figure 5.5 that with SVD, at 100 UEs, the cell is overloaded and the cellular channel is fully utilized.

Figure 5.6 shows the average data rate per user versus the number of UEs in the network for SVD, CSVD and DISCS, respectively (steady-state phase). Up to one copy of each piece of a file is cached in a cluster in the case of CSVD and DISCS. As Figure 5.6 shows, CSVD and DISCS provide important performance gains due to the transmission of video segments from the BS and SMs (distributed cache), as opposed to only transmitting video files from one source (the BS). This speeds up the transmission process and increases the average data rate. In SVD, the average data rate decreases faster with increasing the number of UEs.

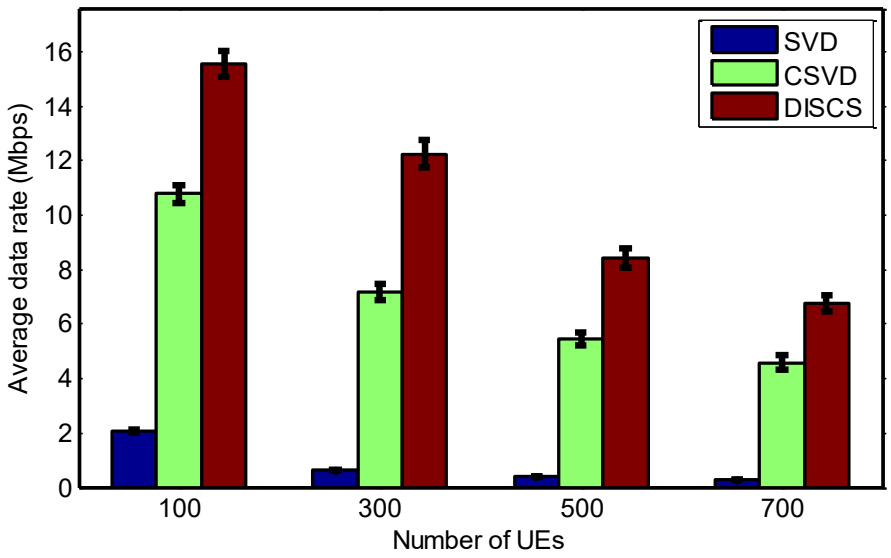

Figure 5.6. Average data rate per user versus number of UEs in the cell (steady-state phase). 2 requests per user and Zipf exponent = 1.5.

For instance, the average data rate decreases from about 2 to 0.63 Mbps when the number of UEs increases from 100 to 300 UEs. This is because the fixed available frequency resources are divided over higher number of UEs. The improvement achieved by the CSVD and DISCS over the SVD increases when the number of UEs increases. This is because increasing the UEs also increases the available SMs and requested and cached files. Thus, more data will be transmitted from the clusters' caches over D2D links rather than being sent from the BS over cellular links. As such, increasing the number of UEs will cause less decrease in the average data rate per user than in the SVD.

Figure 5.6 also shows that DISCS achieves significant improvement over CSVD. This because DISCS speeds up video caching and achieves better hit ratio as discussed in the previous section, which increases the percentage of requests that are satisfied from the cluster's cache and speeds up the transmission. This is also because in the case of DISCS, many files will be sent in parallel from multiple SMs (as opposed to one SM). This

causes further parallelism in sending video files and better load balancing between SMs, which speeds up the transmission of video files and increases the average data rate.

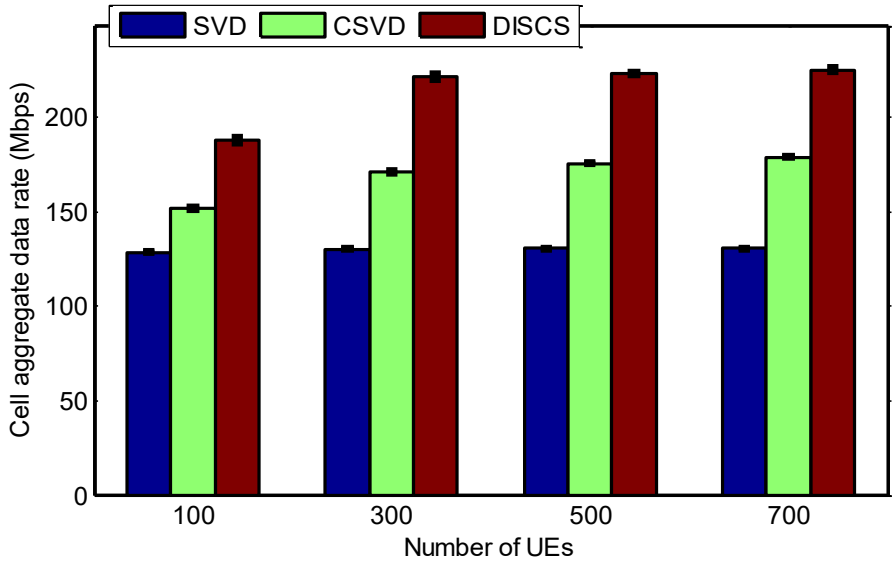

Figure 5.7. Cell's aggregate data rate vs. number of UEs in the cell (transient phase). 2 requests per user and Zipf exponent = 1.5.

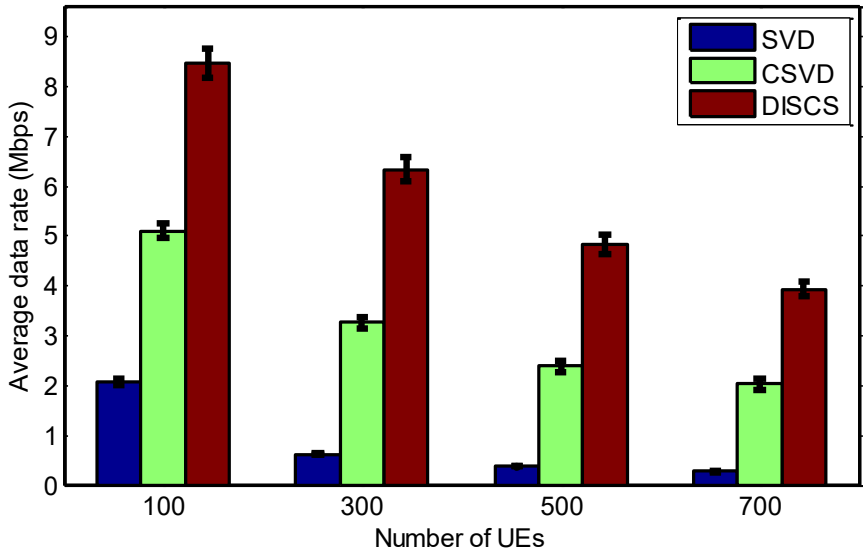

Figure 5.8. Average data rate per user versus number of UEs in the cell (transient phase). 2 requests per user and Zipf exponent = 1.5.

As previously mentioned, the simulations were divided into two phases. The first phase is the transient phase that starts with no video files saved in the distributed caches of the

clusters, and the pieces of the video files accumulate in the distributed caches during this phase as requested by UEs. At the beginning of the steady-state phase, there will by many files in the clusters that were cached during the transient phase. Figures 5.7 and 5.8 show the aggregate data rates and average data rates, respectively, for the transient phase versus the number of UEs. As expected, more improvement is achieved by CSVD and DISCS in the steady-state phase. This is because in the steady-state phase, there are more cached files in the clusters. Hence, more video files will be sent from the distributed cache, which increases the D2D channel utilization and speeds up video transmission. However, good improvements are still achieved by both algorithms over SVD in the transient phase.

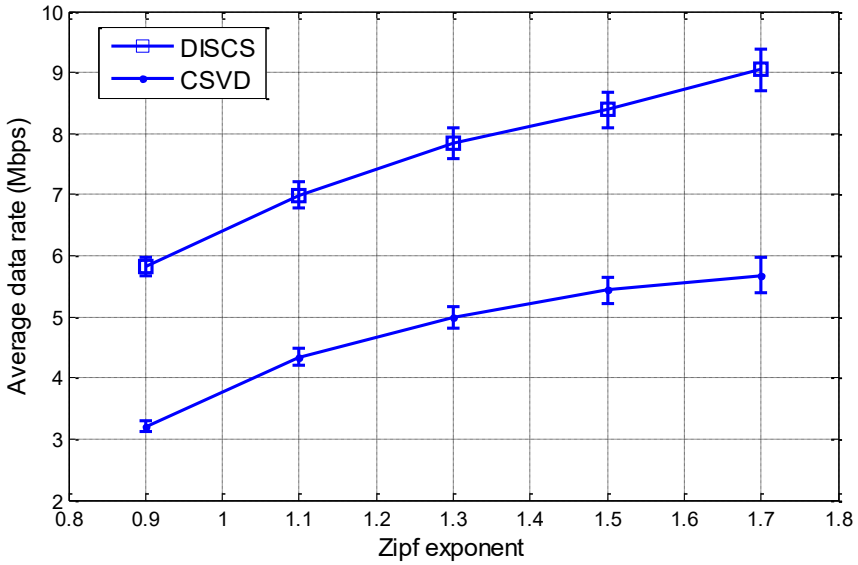

Figure 5.9. Average data rate per user versus Zipf exponent (steady-state phase). 500 UEs in the cell and 2 requests per user.

The impact of the Zipf distribution exponent on the performance of the CSVD and DISCS is shown in Figure 5.9. As can be seen, the average data rate increases by increasing the Zipf exponent. As the popularity of some videos increases, the utilization of the distributed cache increases, because higher percentage of the requests will be found in

the distributed cache and delivered over the D2D channel (rather than the cellular channel). This speeds up the transmission of many videos and increases the average data rate.

Increasing the number of requests made by each UE further increases the utilization of the distributed cache and improves the data rates. The effect of increasing the number of requests can be seen in Table 5.2, which shows the results for the average and aggregate data rates for CSVD and DISCS in the steady-state phase with 2 requests and 3 requests. As can be seen, increasing the number of requests increases the average and aggregate rates for both algorithms. This is because when the number of requests increases, the cached files will be further used by the later requests. Hence, more requests will be satisfied from the distributed cache, which improves the data rates.

Table 5.2. Average and aggregate data rates with 2 and 3 requests.

	CSVD		**DISCS**	
	Average (Mbps)	Aggregate (Mbps)	Average (Mbps)	Aggregate (Mbps)
2 requests	5.43	445.60	8.39	611.24
3 requests	6.05	490.22	9.58	665.29

One can notice that although the algorithms provide significant improvements, they introduce some overhead (Handshake message, Assistance Request message, etc.) needed to implement D2D distribution of video contents. In the following we discuss the overhead of the proposed algorithms. The efficiency is measured as the ratio of transmitted data bits to the transmitted bits (data bits plus transmitted bits in the Handshake message, Assistance Request message, etc.). The overhead equals (1 - efficiency).

With 512 KB piece message, the efficiency of CSVD in the steady-state phase is 0.999936 (overhead of 0.000064) and the efficiency of DISCS is 0.999934 (overhead of

0.000066). The results show that the proposed algorithms have high efficiency (low overhead). The overhead for DISCS is slightly higher than that for CSVD. This is expected as more *Assistance Request* messages are sent by DISCS (due to the DTSMs case).

To improve the average data rates achieved by the CSVD algorithm, more than one copy can be cached of each piece in the cluster. Of course, this would be on the expense of using more of the storage of the SMs. Figure 5.10 shows that an improvement is achieved with the CSVD when the number of cached copies is increased from 1 to 3. This improvement is caused by having the popular files available in more SMs, which allows further parallelism when sending pieces, and allows more load balancing between SMs, which speeds up the transmission of video files to the requesting UEs.

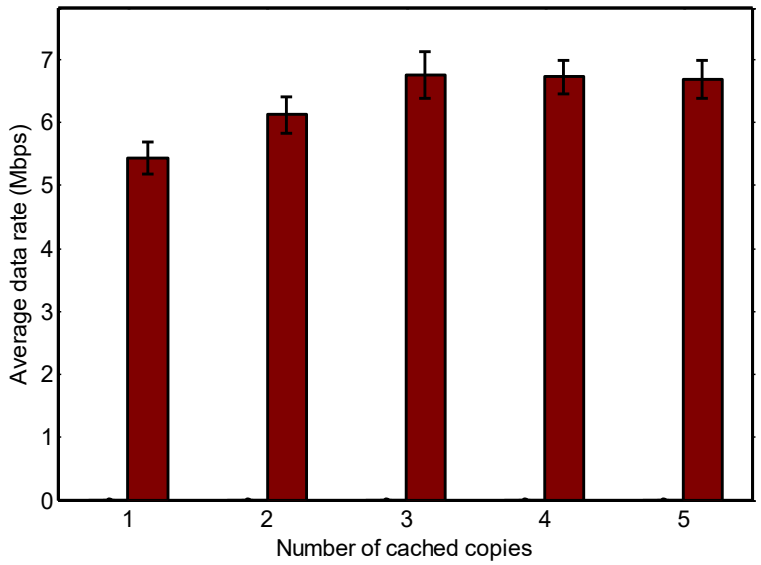

Figure 5.10. Average data rate per user versus number of cached copies (steady-state phase) for CSVD. 500UEs and Zipf exponent = 1.5.

After 3 cached copies, there is no significant effect for the number of cached copies in the cluster, which means that 3 copies in the cluster are enough. This is explained in the

following. The BS sends a piece to an SM directly (STSM) if the number of copies in the cluster is less than the intended number of cached copies. Otherwise, the piece will be sent to the SM from the distributed cache. Hence, increasing the number of cached copies will increase the average data rate up to a certain point. After some point, increasing the number of cached copies beyond a certain value might cause a slight reduction in the average data rate, as too many copies will need to be sent to SMs from the BS over cellular links even when enough copies are already cached in the cluster. This explains why the average data rate for CSVD with 5 cached copies is slightly less than that of CSVD with 3 and 4 cached copies.

5.2.3 The impact of UE mobility

In this section, we analyze the effect of UE mobility. Furthermore, we show how significant performance improvement can still be achieved in the case of UE mobility by employing the proposed improved operation, i.e., requesting all UEs to cache received contents. As previously mentioned, this also increases the fairness among users because it increases the chance of a requesting UE to be a helper/provider in the future.

The probability of movement, P_m, is a parameter that controls the probability that a UE in the simulation is moving. When P_m is 1, all the UEs in the simulation will be moving, and when P_m is 0, all the UEs in the simulation will be stationary. A moving UE will have a random destination in the simulation area. A moving UE walks with a speed of 1.34 m/s which is the average pedestrian speed. When a UE reaches its destination, it generates a new destination and starts moving towards the new destination and so on.

In Addition to considering UE mobility, we updated the channel model to include fast

fading in addition to path loss and shadowing. Rayleigh fading channel model was considered, where the employed Rayleigh fast fading model considers the speed of the UE, the subcarrier frequency, and the number of paths.

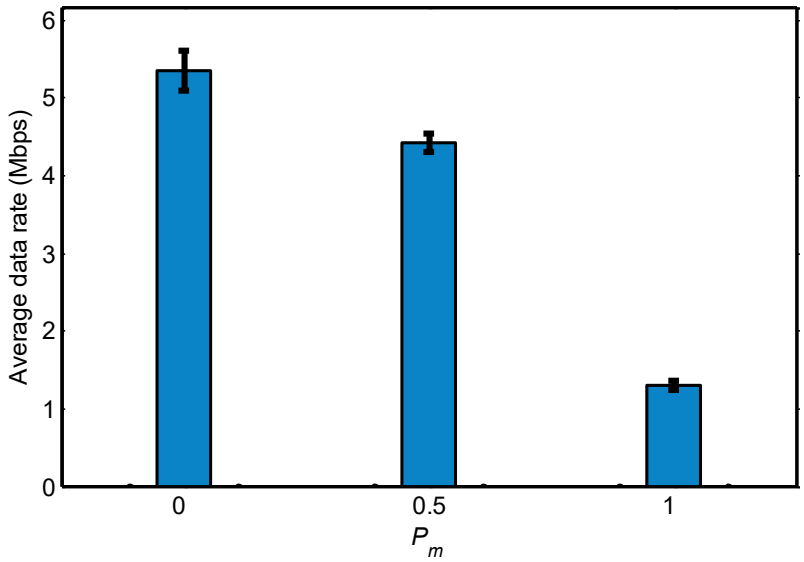

Figure 5.11. Average data rate per user vs. P_m for CSVD (steady-state phase). 500 UEs in the cell and Zipf exponent = 1.5.

Figure 5.11 shows the average data rate per user versus P_m, for CSVD (steady-state phase). As we can see, the average data rate decreases by increasing P_m. When $P_m = 0$, all the UEs in the cell are stationary, which means that UEs with cached contents will be always available as SMs in the central area. However, when P_m increases, some of the UEs will be moving. For instance, at $P_m = 0.5$, half of the UEs in the cell are moving, which means that about half of the SMs in the clusters will be moving as well. As some of these moving SMs (especially the ones with cached popular files) might leave the central area and become NSMs, utilization of cached contents will be reduced. This is because such cached contents will not be available for transmission. This reduces the number of video segments transmitted from the distributed cache, and hence, reduces the

average data rate.

At P_m = 1, all the UEs are moving, which means all SMs are moving. By the time the steady-state phase starts, higher number of SMs would be out of the central area, and hence, their cached contents will not be available for transmission. As such, this further reduces utilization of cached contents, and decreases the average data rate. From the above results, one can see that the movement of UEs has an impact on the performance of proposed algorithms with their conventional operation (especially when all UEs are moving). However, we will see next how the improved operation overcomes this challenge.

In Chapter 3, we discussed an improved operation for the proposed algorithms to overcome the performance degradation that might be caused by the mobility of UEs. With the improved operation, instead of setting the save bit to 1 only in the STSM and DTSMs cases, the save bit will always be set to 1. This means that every UE will be asked to cache received contents. This is because when UEs are moving, any UE could be an SM (after entering the central area), and hence, it would be very beneficial (for the cluster) for any UE that becomes an SM to have cached contents. This also increases the fairness of the algorithms, since in this case UEs that received contents from the clusters' caches could become SMs and provide others with video contents later.

Figure 5.12 shows the average data rate per user for SVD, CSVD, and improved CSVD (iCSVD), with P_m = 1 (all UEs are moving). iCSVD is basically the CSVD with the improved operation where received video segments are always cached. As can be seen in the figure, although UE mobility has reduced the average data rate achieved by CSVD,

CSVD still provides significant improvement over SVD. However, the proposed operation with iCSVD significantly improves the achieved average data rate and overcomes the impact of UE mobility, even at P_m = 1. This is because with the iCSVD, all UEs are caching received video segments. In this case, even if some SMs leave the central area and become NSMs, other UEs with cached contents enter the central area and become SMs with cached contents, which increases the hit ratio and improves the achieved average data rates, when compared to CSVD.

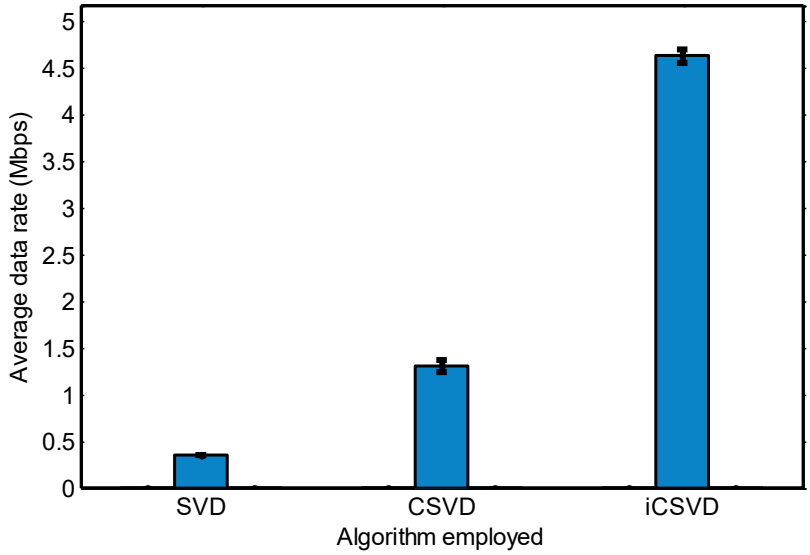

Figure 5.12. Average data rate per user vs. the employed algorithm (steady-state phase). 500 UEs in the cell, Zipf exponent = 1.5, and P_m = 1.

5.3 Summary

In this chapter, we present the results for the hit ratio of the CSVD and DISCS algorithms obtained from both the analytical models and the simulations performed using the DEVS model. The simulation results are very close to the analytical results, and both show that DISCS improve the hit ratio over CSVD. Moreover, we discuss simulation results for the DL cell's aggregate data rate and average data rate per user. Results show that employing

CSVD and DISCS provides significant improvements over conventional transmission methods in terms of the studied data rates. Furthermore, Results show that DISCS achieves significant improvements over CSVD. Finally, we study the effect of UE mobility and show that significant performance improvement can still be achieved in the case of UE mobility by employing the proposed improved operation, i.e., requesting all UEs to cache received contents.

Chapter 6

Analyzing the effect of LTE-A transmission parameters on video streaming QoE

In this chapter, we study the impact of transmission parameters in cellular networks on multiple video streaming QoE metrics. We consider DASH-based video streaming [24], [25]. This is because DASH has been employed by big video streaming platforms, such as YouTube and Netflix, and it is being adopted by an increasing number of video applications.

To the best of our knowledge, this is the first work that considers the impact of the transmission parameters at both the cell level and link level on DASH-based video streaming QoE. Studying the impact of the transmission conditions at the cellular link level on the QoE is important. This is because understanding this impact helps cellular network service providers reduce the risk of having dissatisfied customers by, for instance, determining the link-level data rates required to achieve a certain level of QoE. However, the main contribution here is the study of the impact of the transmission parameters at the cell level in addition to the link-level parameters. This is crucial because such study at both levels provides a valuable insight for LTE-A network design, especially in terms of small cells deployment and configuration in urban areas (where there is usually a high density of users). As video streaming is continuously gaining in popularity nowadays, supporting good QoE video service is an important reason for customers to choose a cellular provider.

We built a model for an LTE-A network, and used the model to run various simulations under different scenarios. We present an exploratory data analysis of the results to provide a quantitative evaluation of the effect of the transmission parameters on many objective QoE metrics, such as video stalling and initial delay.

6.1 Modeling video streaming over an LTE-A network

6.1.1 DEVS model

We extended the DEVS model presented in Chapter 4 to simulate video streaming over an LTE-A network and employed the model to study video streaming QoE. The DEVS model is presented in Figure 6.1. As we can see, we defined a *Cell* coupled model that contains a *BS*, a Transmission *Medium*, and many *UE* coupled models. It also contains a *Cell Manager* and *Log Manager* atomic models. The *BS* coupled model includes four atomic models: *BS Queue*, *BS Controller*, *Scheduler*, and *Transmitter*. Received messages are buffered at the *BS Queue*. The *BS Controller* controls the various components of the BS (queue, scheduler, etc.). The *Scheduler* is responsible for allocation of frequency resources in the next TTI to active users. Every TTI, the *BS Controller* asks the *Transmitter* to send the data that was scheduled for transmission during this TTI. A *UE* coupled model contains four atomic models: *UE Queue*, *UE Controller*, *Streaming Client*, and *DASH controller*. Messages received are buffered at the *UE Queue*. The *UE Controller* controls the different components of the UE.

The DASH-based streaming client is implemented in the *Streaming Client* and *DASH Controller* atomic models. The streaming client manages the video buffer. It adds received video segments to the video buffer and removes video segments that were

played from the buffer. As the video buffer usually has a certain length (could be shorter than the video), it is implemented as a sliding window; video segments played are removed from the buffer, which slides to cover the next segments in the stream. The DASH controller implements the video bit rate adaptation algorithm (Section 6.1.2). It monitors the video playout buffer, and updates the video bit rate accordingly.

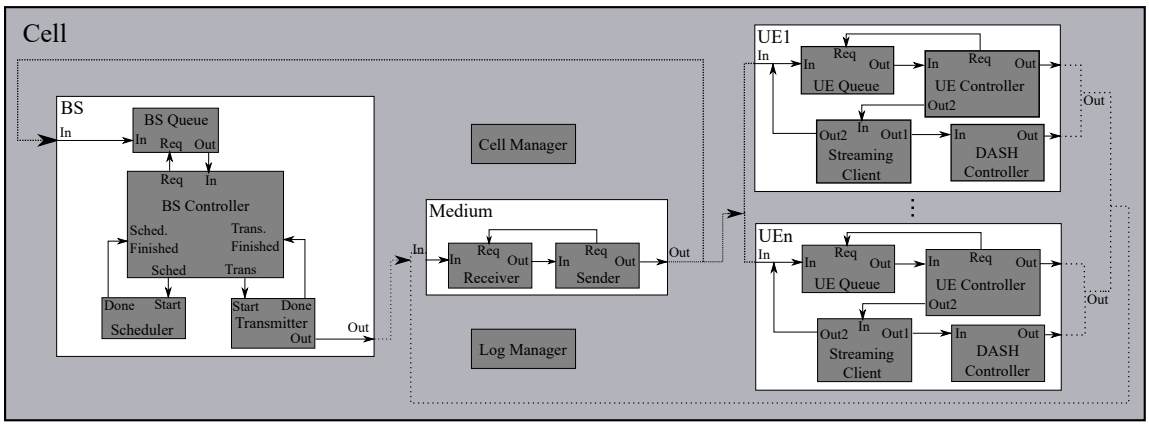

Figure 6.1. Coupled DEVS model of the LTE-A network.

The *Medium* model simulates the transmission medium and the *Cell Manager* atomic model initializes and sets the parameters of the cellular DLs and ULs between the BS and the UEs. The *Log Manager* logs simulation events and record statistics during the simulations.

In addition to the atomic models above, many other passive classes where developed to model other components of the system such as classes to model the cellular links, download sessions the BS has with UEs, etc.

6.1.2 DASH controller

The adaptation algorithm of DASH is the most important part at the client's side [40], [41], [46]. We refer to the component of the video client that runs the adaptation

algorithm as the DASH controller. Here, we use the buffer-based approach proposed in [41], as it is robust against throughput variation in wireless environments. The adaptation algorithm we use is a piecewise function, *f(L)*, that uses the length of the playout buffer, *L*, to determine the video bit rate as shown in Figure 6.2.

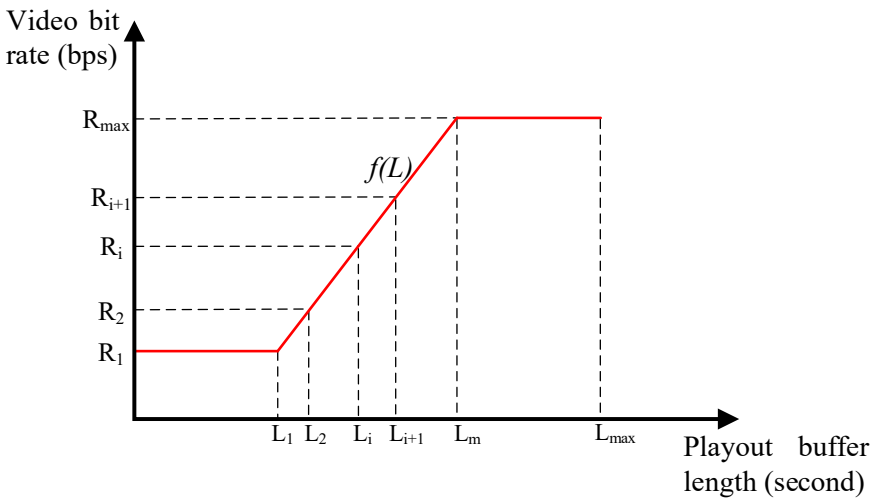

Figure 6.2. Buffer-based adaptation algorithm (adopted from [41]**).**

At the beginning when the buffer is empty, the video bit rate will be set to the minimum level, R_{min}. When the playout buffer length reaches a certain value (L_1 in Figure 6.2), the client will ask for next higher video bit rate (R_2 in Figure 6.2). The algorithm follows a simple rule, stay at rate R_{i+1} as long as the playout buffer length is between L_i and L_{i+1}.

6.2 Verification and validation of the model

Our video streaming DEVS model was built on top of our cellular network DEVS model. V&V of the cellular network model is discussed in Chapter 4. Here, we discuss the approaches used for V&V of the video streaming model and present some of the validation tests that we carried out to ensure the correctness of this model. For model verification, we performed thorough debugging, structured walk-through, and step-by-

step analysis on this model in a bottom-up fashion. Each atomic model is verified via thorough debugging, structured walk-through, and step-by-step testing. After verification of all the atomic models in a coupled model, the coupled model itself is tested. Similarly, after verification of all the atomic and coupled submodels in a coupled model, the integration of the submodels is tested. This thorough testing is performed for every atomic and coupled submodel in a bottom-up fashion until the top-level coupled model is verified.

Regarding model validation, we used the popular three-step validation approach based on the work of Naylor and Finger [89]. To carry out the steps in this approach, we executed various subjective and objective evaluations. In the following, we present how each one of these steps was implemented.

6.2.1 Face validation

Our video streaming DEVS model was developed in collaboration with Professor Stenio Fernandes from the Federal University of Pernambuco. Dr. Fernandes is an expert in video streaming modeling and simulation, and he has publications in the field (e.g., [94]). During the implementation of the DEVS model, Dr. Fernandes was a visiting professor at Carleton University, where we used to have regular discussions on the conceptual model, the assumptions and simplifications adopted, the behaviour of the model, as well as the obtained simulation results.

6.2.2 Validation of model assumptions

The conceptual model and assumptions employed in our video streaming model are discussed in Section 2.1.2 and Section 6.1. All these assumptions are standard and

common assumptions about the operation and dynamics of HTTP video streaming that are adapted by many researchers in the filed of video streaming and obtained from trusted sources (e.g., [41]). This includes both structural and data assumptions.

6.2.3 Validation of input/output transformations

In this stage of model validation, the model is treated as input/output transformations. As with the previous model, we performed multiple tests to evaluate the input/output transformations of our model and to ensure its operational validity [88]. Regarding the initial delay, it is basically the delay from the time a video streaming request is sent until the time the last video segment (e.g., the fourth segment) to start playout is received. This measurement is based on the cellular network model (validation of this model is discussed in Chapter 4). Furthermore, the model was traced (followed) to ensure that the start and end of initial delay are taken at these events. As such, we focus more on validating the results for the number of rebufferings and video bit rate in this section. In the following, we present the performed tests.

1) Fixed values test

In this test, the input variables and the system parameters are set to constant values. The computerized model is executed, and the output is compared to expected values. We ran various simulations with our video streaming model, where the video bit rate was fixed to the lowest level (384 kbps) and the number of UEs in the cell was set to a low value (10 UEs) that are placed at fixed distance from the BS (100 meters). The average number of rebufferings is measured and results have shown that under such setup the average number of rebufferings is always 0. This is expected because in such case, the average data rate experienced by UEs is higher than the video bit rate. As such, video playout

buffer will not be depleted, and hence, rebufferings will not be experienced.

Various simulations were also performed to validate the results for the video bit rate. For instance, an experiment was conducted with 10 UEs in the cell placed at fixed distance from the BS (100 meters). Video segments were available in 384 kbps and 768 kbps. Playout was set to start after receiving the second video segment (segment length is 10 seconds) and the DASH controller was set to update the video bit rate to 768 kbps when the playout buffer length is equal to or greater than 10 seconds. As expected, the first segment was received with the lowest video bit rate, while the rest of the video segments were received with 768 kbps. As the data rate is much higher than 768 kbps, it can continuously support downloading video segments at this video bit rate without depleting the playout buffer length and reducing the video bit rate to 384 kbps.

2) Degenerate test

Many degenerate tests were performed on the video streaming model to validate its behavior. Experiments were conducted to study the impact of increasing the number of users or decreasing the channel bandwidth on video streaming QoE metrics. The tests show that the model produces the expected input/output transformations. For instance, increasing the number of UEs in the cell increases the number of rebufferings and initial delays experienced by users. Similarly, decreasing the channel bandwidth increases the number of rebufferings and initial delays experienced by users in the cell. All these results and others are discussed in detail in Section 5.3.

3) Internal validity test

Internal validity is concerned with the variability of the model to make sure that the

produced results from different simulations are consistent. Many simulation runs were executed with the video streaming DEVS model to ensure its internal validity. The obtained results from different simulation runs show consistency and have an acceptable rang of variability. This can be inferred from the MoE values for 95% confidence interval that are computed and shown for all simulation results obtained from the DEVS model.

4) Predictive validation and event validity

As previously mentioned. The video streaming model is built on top of our cellular network DEVS model. The most complex aspect of this model is the client which handles the video playout buffer and triggering of playback and rebuffering events. In this section, we use two objective approaches for validation; predictive validation and event validity.

With predictive validation, the simulation model is executed in a certain scenario to predict the behavior of the real system. The simulation results are then compared to the behavior of the real system to check if both exhibit a similar behavior. With event validity, the model is traced to check the occurrence of certain events. These occurrences are compared to those of the real system to check if events in the model and the system take place in a similar way.

Here, we show some experiments we performed to test input/output transformations of our model. These tests provide both predictive and event validation. In these experiments, we set the values of some input variables and parameters, such as the number of UEs in the cell and the available video bit rate. Then, we check the value of the average data rate. Based on the value of the average data rate and the used video bit rate we decide if the users in a real video streaming scenario would experience rebufferings or not. We

compare the output (number of rebufferings) produced by the simulation model and that of a real system.

In the conducted experiments, all the UEs are located at a fixed distance from the BS (300 meter). The available video bit rate, playout buffer length to start playback, and the length of videos are set to fixed values. Each UE makes one request for a video and finishes playback of the video. All the UEs make the requests at the same time (at the beginning of the simulation). For each experiment, we present the simulation setup. Then, we discus the behavior of a real streaming system in that scenario. Thereafter, we analyze the average data rate and the number of rebuffering results exhibited by the simulation model.

In the first experiment, we set the available video bit rate to 384 kbps, the length of a video to 100 seconds (10 segments), and the length of video playout buffer to start playout to 10 seconds (1 segment). In such system, video playback starts after a video segment is received. When video playout starts, the playout buffer will be consumed at a rate of 384 kbps. The next video segment to play should be received before the currently buffered segment is consumed. In order for this condition to be satisfied, the data rate should be higher than the video bit rate of the currently played segment (rate at which the current video segment is consumed). If this condition is satisfied, the system will start playout of the next segment after 10 seconds (consumption of the current segment). The third segment should be received before playout of the second segment is finished, and so on. Therefore, if the data rate is higher than the video bit rate, playback will continue smoothly without any rebufferings. However, if the data rate is lower than the video bit rate, rebuffering will occur right after playout of each segment, as the next segment will not be available. As such, each UE will experience 9 rebufferings. We ran various

simulations with different number of UEs in the cell to control the average data rate and analyze the number of rebufferings experienced by each UE in the cell. As mentioned above, all UEs are placed at the same distance from the BS and cellular resources are allocated in a round robin fashion. As such, UEs should experience close values for the average data rate, and hence, experience the same number of rebufferings. Simulation results have shown that at 350 UEs in the cell, the average data rate is 397.80 kbps which is higher than the video bit rate. In this case, all the UEs experience zero rebufferings which is the same behavior of a real video streaming system. At 400 UEs in the cell, the average data rate is 348.08 kbps, which is lower than the video bit rate. As expected, all UEs in the cell experienced 9 rebufferings. The same experiments were conducted for different values of the video bit rate. At 768 kbps, the average data rate at 150 UEs is 924.02 kbps which is higher than the video bit rate, and hence, all UEs in the cell experienced zero rebufferings. At 200 UEs, however, the average data rate is 693.12 kbps which is lower than the video bit rate. All UEs in this case experienced 9 rebufferings.

The results above provide predictive validation as the simulation model behaved in the same way as the real system (no rebufferings were experienced when the data rate is higher than the video bit rate). Furthermore, the results provide event validation because the number and occurrence of rebufferings match the expected results.

6.3 Simulation scenarios and results

We executed simulations to evaluate video streaming QoE for users in terms of the metrics presented in Chapter 2. Table 6.1 shows the simulation setup. The simulation parameters are taken from the LTE-A standard [86]. The simulations consider a single

LTE-A cell with 100 to 500 UEs. The urban macro propagation model [86] was used for cellular links with a DL operating carrier frequency of 900 MHz.

Table 6.1. Simulation setup.

Parameter	Value
Cellular channel BW (MHz)	5, 10, 20
Cell range (m)	500
BS antenna gain (dB)	12
BS transmission power (dBm)	23, 33, 43
UE antenna gain (dB)	0
UE transmission power (dBm)	21
Noise spectral density (dBm)	-174
Antenna height (m)	15
Transmission model	UTRA-FDD
DL carrier frequency	900MHz
Number of requests by a UE	2
Area configuration	Urban
UE receiver noise figure (dB)	9
Segment length (s)	10
Number of buffered segments to start playout	4
Video bit rate levels (kbps)	384, 768, 2000, 4000
Videos length (s)	441

Table 6.2. Simulation scenarios.

Scenario	Number of UEs	Cell bandwidth (MHz)	BS transmission power (dBm)
Scenario 1	100	10	43
Scenario 2	300	10	43
Scenario 3	500	10	43
Scenario 4	500	5	43
Scenario 5	500	20	43
Scenario 6	500	10	23
Scenario 7	500	10	33

Various simulation scenarios were considered. Table 6.2 shows the setup considered in each scenario. In Group 1 (the first 3 scenarios), the cell bandwidth and BS transmission

power are fixed to 10 MHz and 43 dBm, respectively, and the number of UEs in the cell is variable. In Group 2 (scenarios 3, 4, and 5), the number of UEs and BS transmission power are set to 500 and 43 dBm, respectively, and the cell bandwidth is variable. In Group 3 (scenarios 3, 6, and 7), the number of UEs and the cell bandwidth are fixed, and the BS transmission power is variable.

In each iteration of the simulation, the UEs are randomly distributed throughout the cell according to a uniform distribution. The UEs then start requesting video streams. During each iteration of the simulation, each UE will request two video streams. A UE requests a video stream, and after finishing the playout, it will request a second video. The arrival of requests is generated according a Poisson arrival process. The length of the videos is 441 s, which is the mean length of a YouTube video [93]. Four video bit rate levels were used as shown in Table 6.1. These are adapted from the H.264/AVC video coding standard [95].

We measured the number of rebufferings, video continuity index, initial delay, and video bit rate levels of the received video segments. Table 6.3 shows the average values for the first three QoE metrics, along with the MoE values for a 95% confidence interval.

Table 6.3 shows that the worst results are for scenarios 4 and 6. This is expected, as scenario 4 has the lowest cell bandwidth (5 MHz) and scenario 6 has the lowest BS transmission power (23 dBm). As the cell bandwidth decreases, the transmission rate decreases, and this will have a significant impact on the QoE metrics. When the transmission rate decreases, the time to transmit video segments to the UEs will increase.

This increase in the delay will increase the possibility of video buffer depletion and consequently increase the number of rebufferings. Similarly, when the BS transmission power decreases, the received power at the UE decreases. This will consequently decrease the transmission rate and increase the number of rebufferings.

Table 6.3. Simulation results.

Scenario	Number of rebufferings		Cont. index		Initial delay (sec)	
	Mean	MoE	Mean	MoE	Mean	MoE
Scenario 1	0	0	1	0	15.274	0.4836
Scenario 2	0	0	1	0	34.937	0.5920
Scenario 3	3.4118	0.0216	0.7461	0.0014	57.132	0.6126
Scenario 4	7.8300	0.0104	0.3999	0.0004	92.667	1.0136
Scenario 5	0	0	1	0	31.693	0.4147
Scenario 6	7.8020	0.0110	0.4062	0.0004	90.988	1.0050
Scenario 7	5.3156	0.0147	0.5817	0.0005	66.372	0.7064

The results for the continuity index agree with the results for the number of rebufferings. The continuity index values are the lowest for scenarios 4 and 6. As the bandwidth or the BS transmission power decreases, the transmission rate decreases. In addition to increasing the number of possible rebufferings, this also increases the rebuffering time (the time needed to receive 4 video segments to start playout) in the case of rebuffering, which decreases the continuity index. The results for initial delay also match these for the number of rebufferings. Scenarios 4 and 6 have the highest values for the initial delay. This can be explained similarly. As the transmission rate decreases, the transmission time of segments increases, which increases the time to start playout and increases the initial delay.

Table 6.3 shows that the best results were obtained by scenario 1. This is expected, as scenario 1 has the lowest number of UEs (100) sharing a 10 MHz bandwidth and 43 dBm

transmission power. As the number of UEs decreases, more resources will be available for each UE. This increases the average data rate per user and speeds up the transmission of video segments to the UEs. This decreases the number of rebufferings, rebuffering time, and initial delay. Table 6.3 shows that with scenarios 1, 2, and 5, the number of rebufferings is 0 for all the video streams. This is because in these scenarios there is either a relatively low number of UEs (scenarios 1 and 2) or high available cell bandwidth (scenario 5). This explains why the MoE is 0 for these measurements. The same can be noticed for the continuity index of scenarios 1, 2, and 5. As all the video streams have a continuity index of 1, the MoE for these measurements is 0.

Table 6.4 shows the average video bit rate results for all the scenarios. The results for the average video bit rate also match the results in Table 6.3. Table 6.4 shows that for scenarios 4 and 6, the average video bit rate is 384 kbps. This means that all the video segments in these scenarios were requested with the lowest video bit rate, which is the worst case scenario. This also explains why the MoE is 0 for these mean values.

Table 6.4. Average video bit rates.

Scenario	Average video bit rate (kbps)	
	Mean	MoE
Scenario 1 (100, 10, 43)	1039.3	4.5225
Scenario 2 (300, 10, 43)	445.64	1.2448
Scenario 3 (500, 10, 43)	397.17	0.2888
Scenario 4 (500, 5, 43)	384.00	0
Scenario 5 (500, 20, 43)	482.04	1.0089
Scenario 6 (500, 10, 23)	384.00	0
Scenario 7 (500, 10, 33)	385.68	0.0376

The highest video bit rate (1039.3 kbps) was achieved in scenario 1. DASH adapts the

video bit rate according to the available throughput or buffer size. As we use a buffer-based adaptation algorithm, the playout buffer length will be longer when there is a high transmission rate from the BS (as in scenario 1), and accordingly, the requested bit rate will be higher. On the other hand, when the transmission rate is low (due to low available bandwidth, high number of UEs, or low transmission power) the playout buffer will be smaller, and the requested video bit rate will be lower.

We can see that in scenarios 2 to 7, the average video bit rate is the least affected metric. This is because with DASH, video bit rate is the last metric it tends to improve. When the available bandwidth is enough to keep the playout buffer consistently high, DASH will keep asking for a higher video bit rate, which significantly increases the average video bit rate. This is the case in scenario 1 where there are only 100 UEs in the cell sharing a 10 MHz bandwidth.

In the following, we will analyze the effect of each one of these transmission parameters on the QoE metrics.

6.3.1 Number of UEs in the cell

Figure 6.3 shows the Empirical Cumulative Distribution Function (ECDF) of the initial delay with 100, 300, and 500 UEs in the cell. The ECDF value shows the percentage of values that are less than or equal to a given initial delay. For example, the ECDF for 100 UEs shows that 28% of the streams have initial delay of 10s or less. We can see that the number of users in the cell has an impact on the initial delay. For instance, with 100 UEs, all the requests have an initial delay of 36s or less, while with 300 UEs, only 46% of the video streams have an initial delay of 36s or less, and all the other requests have higher

initial delay. Results for initial delay are even worse for 500 UEs. Only 17% of the requests have initial delay of 36s or less. As the number of UEs increases, the cell bandwidth will be shared by more UEs, which reduces the average data rate per UE. This increases the transmission delay of video segments, and consequently increases the initial delay (time to receive the first 4 video segments and start playout).

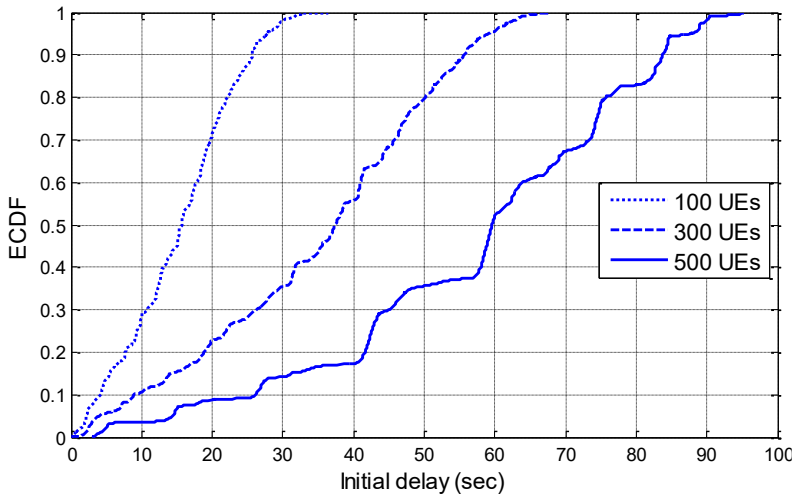

Figure 6.3. The ECDF of the initial delay with different number of UEs in the cell. Bandwidth = 10 MHz and BS transmission power = 43 dBm.

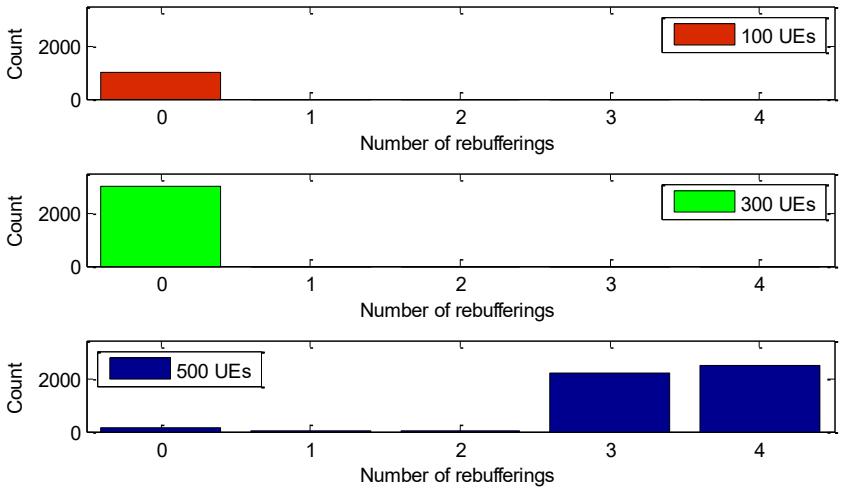

Figure 6.4. The histogram of the number of rebufferings with different number of UEs in the cell. Bandwidth = 10 MHz and BS transmission power = 43 dBm.

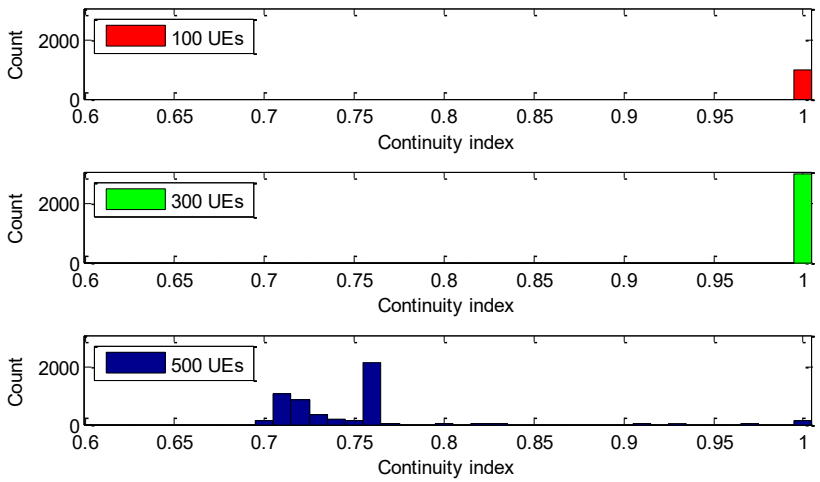

Figure 6.5. The histogram of the continuity index with different number of UEs in the cell. Bandwidth = 10 MHz and BS transmission power = 43 dBm.

Figure 6.4 shows the histogram of the number of rebufferings with 100, 300, and 500 UE in the cell. As with the initial delay, there is a clear effect for the number of UEs on the number of rebufferings. All the video streams with 100 UEs have 0 rebufferings. With 300 UEs, the bandwidth is still enough to avoid rebufferings for all the video streams. With 500 UEs, the effect increasing the number of UEs in the cell starts to show, and about 96% of the video streams have 3 or 4 rebufferings.

Figure 6.5 shows the histogram of the continuity index with 100, 300, and 500 UEs in the cell. With 100 and 300 UEs, the continuity index for all the streams is 1 (0 rebufferings). With 500 UEs, about 96% of the streams have continuity indices between 0.7 and .77. The effect of the number of UEs on the number of rebufferings and continuity index can be explained similarly. Increasing the number of UEs in the cell decreases the average data rate per user. This increases the transmission delay for video segments, which increases the possibility of video buffer depletion, and increases the number rebufferings. This also increases rebuffering time, resulting in a lower continuity index.

6.3.2 Cell bandwidth

Figure 6.6 shows the ECDF of the initial delay with cell bandwidth of 5, 10, and 20 MHz. We can see that there is a clear effect for the channel bandwidth of the cell on the initial delay. With a 20 MHz channel, all the video streams have an initial delay of 62s or less. With 10 MHz channel however, only 52% of the video streams have initial delay of 62s or less, and with 5 MHz, only 20% of the video streams have initial delay of 62s or less. Reducing the cell bandwidth reduces the average data rate, which results in a higher initial delay values for the UEs in the cell. The decrease in the average data rate increases the transmission time of video segments, as mentioned above, which increases the possibility of video buffer depletion, and increases the number rebufferings. This also increases rebuffering time, resulting in a lower continuity index.

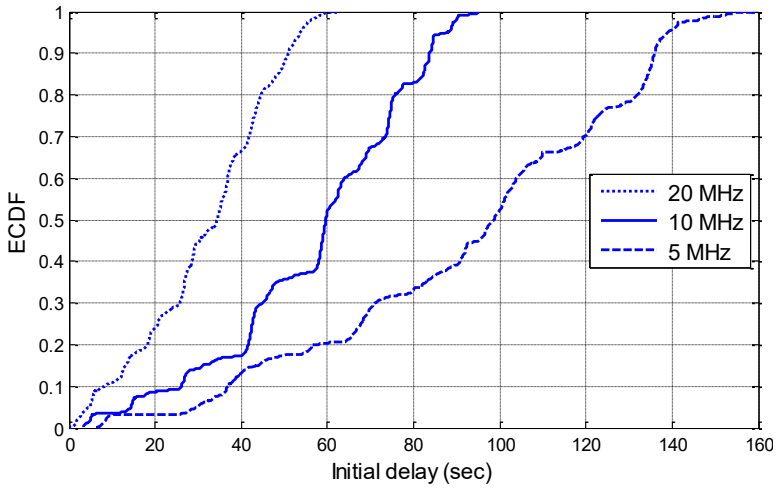

Figure 6.6. The ECDF of the initial delay for different values of cell bandwidth. Number of UEs in the cell = 500 and BS transmission power = 43 dBm.

Figure 6.7 shows the histogram of the number of rebufferings with 500 UEs, and different values of the cell bandwidth. With 5 MHz channel, about 83% of the video streams have 8 rebuffering events and the remaining streams have 7 rebuffering events. With 10 MHz channel, about 96% of the streams have 3 or 4 rebufferings, and the

remaining streams have 0, 1, or 2 rebufferings. Increasing the channel bandwidth from 5 MHz to 10 MHz significantly improves the QoE for the users in the cell in terms of rebufferings. When the bandwidth is increased to 20 MHz, all the streams have 0 rebufferings.

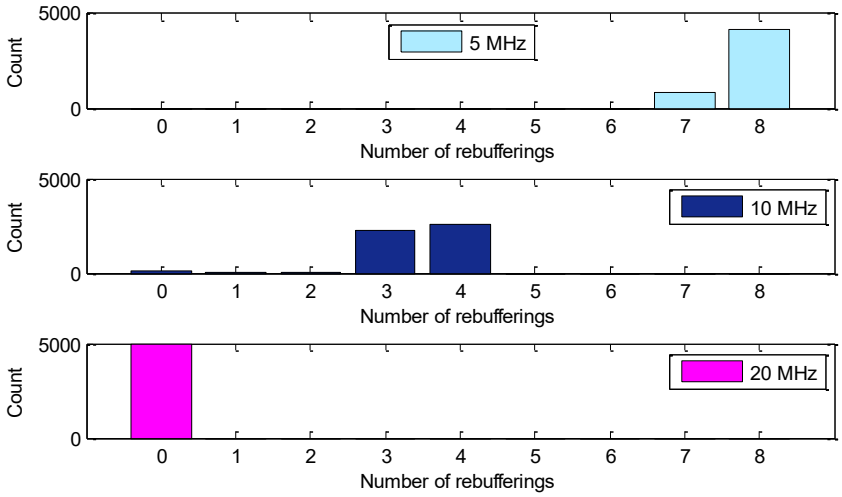

Figure 6.7. The histogram of the number of rebufferings with different values of cell bandwidth. Number of UEs in the cell = 500 and BS transmission power = 43 dBm.

Figure 6.8 shows the histogram of the continuity index with 500 UEs, and different values of the cell bandwidth. With a 5 MHz channel, the continuity index of the video streams is between 0.38 and 0.46, which is very low. With a 10 MHz channel, about 96% of the streams have continuity indices between 0.70 and 0.77, and all the video streams with 20 MHz channel have a continuity index of 1.

6.3.3 BS transmission power

Figure 6.9 shows the ECDF of the initial delay with different values for the BS transmission power. It is worth mentioning that in LTE-A networks, there are different mechanisms the BS might use for power control. However, in this study, we assume the BS transmits at a certain power level in every scenario, to study the effect of the BS

transmission power. As can be seen in Figure 6.9, there is a considerable impact for the BS transmission power on the initial delay. As previously explained, this is because the BS transmission power affects the received power, which affects the transmission rate.

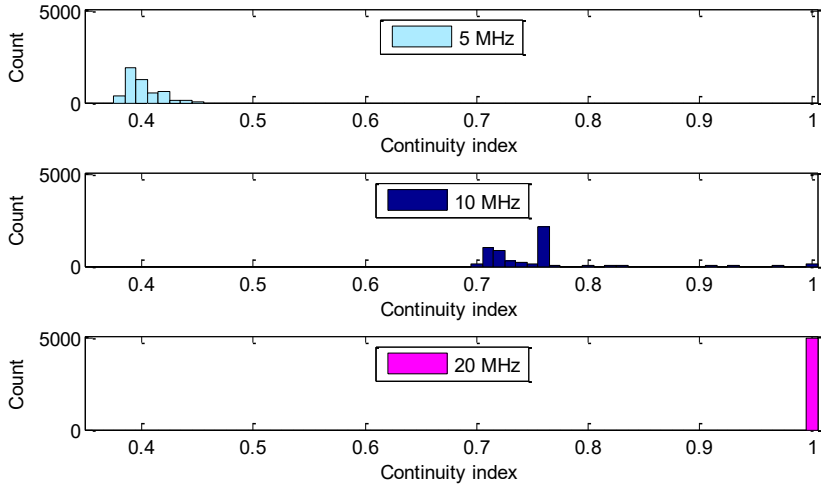

Figure 6.8. The Histogram of the continuity index with different cell bandwidth. Number of UEs in the cell = 500 and BS transmission power = 43 dBm.

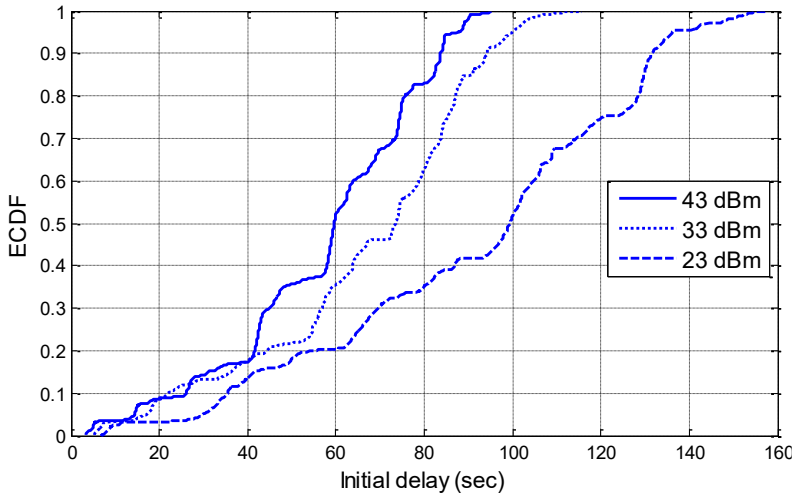

Figure 6.9. The ECDF of the initial delay for different values of BS transmission power. Number of UEs in the cell = 500 and bandwidth = 10 MHz.

Figure 6.10 shows the histogram of the number of rebufferings with 500 UEs, and different values of the BS transmission power. At 23 dBm, it can be seen that 80% of the streams have 8 rebufferings and the rest have 7 rebufferings. When the BS transmission power is increased to 33

dBm, about 96% of the streams have 5 and 6 rebufferings, and some streams have 4 rebufferings. This is still considered an improvement when compared to the results with 23 dBm. At 43 dBm, about 96% of the streams have 3 or 4 rebufferings and the rest have 0, 1, or 2 rebufferings.

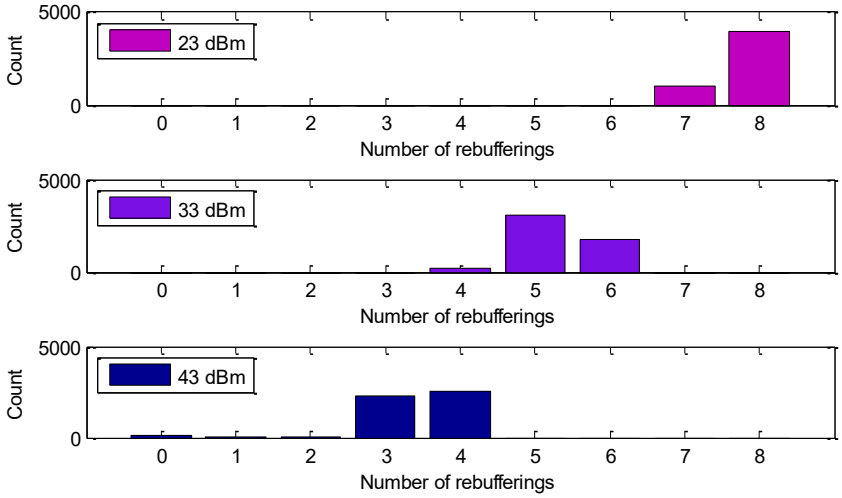

Figure 6.10. The histogram of the number of rebufferings with different values of BS transmission power. Number of UEs = 500 and bandwidth = 10 MHz.

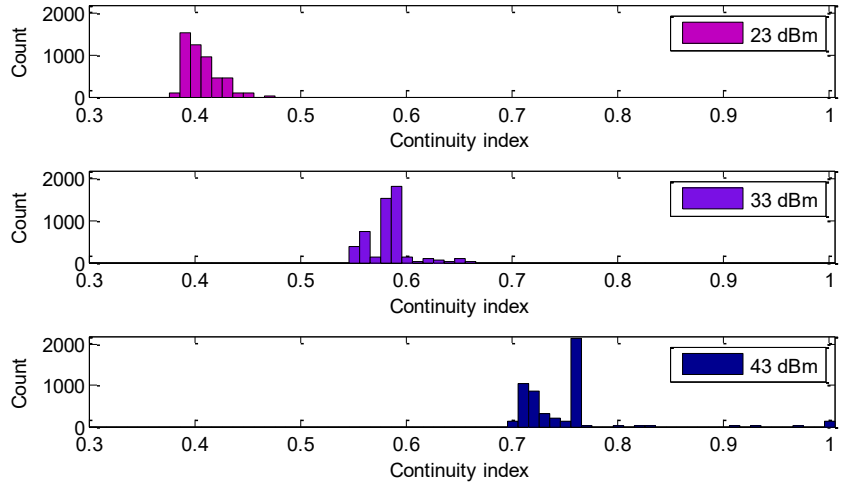

Figure 6.11. The histogram of the continuity index with different values of BS transmission power. Number of UEs = 500 and bandwidth = 10 MHz.

Figure 6.11 shows the histogram of the continuity index with 500 UEs, and different values of the BS transmission power. At 23 dBm, the continuity indices of the streams are between 0.38 and .46, which is very low. Increasing the transmission power to 33

dBm clearly increases the continuity index range of the streams. At 33 dBm, the continuity indices of the streams are between 0.55 and 0.66. At 43 dBm, about 96% of them are between 0.7 and 0.77, as mentioned before. From Figures 6.9, 6.10, and 6.11, it can be concluded that decreasing the BS transmission power decreases the received signal at the UE. This decreases the data rate, and consequently results in higher number of rebufferings, higher initial delay, and lower continuity index.

6.3.4 Distance between the BS and UE

To study the effect of distance between the BS and the UE, we ran simulations with 3 more scenarios. In these scenarios, the number of UEs is 500, cell bandwidth is 10MHz, and the BS transmission power is 43 dBm. In each one of these simulations, all the UEs are placed in a circle with fixed distance from the BS. The distances used are 100m, 300m, and 500m. This will show the impact of the transmission distance between the UE and BS on the QoE. This also gives an indication about the effect of the cell range on the QoE of UEs in the cell.

Table 6.5. Simulation results for the impact of distance.

Distance (meter)	**Number of rebufferings**		**Cont. index**		**Initial delay (sec)**		**Average video bit rate (kbps)**	
	Mean	MoE	Mean	MoE	Mean	MoE	Mean	MoE
100	0.032	0.0049	0.9979	0.0003	38.24	0.4945	416.2	0.8045
300	2.643	0.0190	0.7995	0.0013	50.47	0.5819	395.6	0.2281
500	4.458	0.0202	0.6728	0.0012	59.01	0.6535	390.9	0.1384

Table 6.5 shows the average values for the QoE metrics for each one of these scenarios along with the MoE for 95% confidence interval. As can be seen, there is a clear impact for the distance between the BS and UE on all the QoE metrics. As the distance increases,

the path loss increases, which reduces the received signal power at the UE, and consequently reduces the data rate. This degrades the QoE as previously discussed.

6.4 Summary

In this chapter, we study the impact of cellular transmission parameters on the QoE metrics of DASH-based video streaming. We consider parameters at both the cell and link levels. At the cell level, we studied the cell channel bandwidth and the number of UEs in the cell. As both of these parameters control the bandwidth a UE shares, and consequently controls the average data rate, they have a significant impact on all the studied QoE metrics. At the link level, we study the impact of the BS transmission power and the BS-UE distance. Results show that both of these parameters have a considerable impact on all the studied QoE metrics. In most of the studied scenarios, the average video bit rate was the least affected metric by the studied parameters. This is because video bit rate is the last metric that DASH tends to improve. As such, the exploratory data analysis did not just quantify some intuitive behavior, but also brought some new findings to discuss.

Chapter 7

Progressive caching with DASH-based and BS-assisted D2D video streaming in cellular networks

As previously discussed, with the continuously increasing popularity of video streaming applications, supporting video streaming services with high QoE is becoming a main concern for cellular network operators. In this chapter, we propose and evaluate an architecture, namely DABAST, that improves the QoE of video streaming in cellular networks. The proposed architecture employs the following techniques:

- The CSVD and DISCS algorithms (discussed in Chapter 3 and Chapter 5). Recall that CSVD and DISCS implement progressive caching of video contents and D2D communication. This provides BS-assisted P2P video content transmission in cellular networks. Here, CSVD and DISCS are adapted in the context of video streaming. We evaluate the proposed architecture in terms of video streaming QoE metrics.
- The proposed architecture also supports DASH-based video streaming. DASH is employed by the proposed architecture due to its advantages and to allow support for DASH-based applications.

With DABAST, employing CSVD and DISCS helps relaxing the RAN bottleneck, which is the main bottleneck in cellular networks with limited frequency resources that are shared among a large number of users. In this chapter, we discuss the structure of

DABAST and its operation.

We also use the DEVS formalisms to build a model for the proposed architecture in an LTE-A network. Simulations based on this model are used to evaluate the performance of DABAST. We first consider DABAST which employs CSVD. We provide a thorough analysis of the performance improvements achieved by DABAST in terms of many video streaming QoE metrics. Furthermore, we investigate the performance of DABAST in many scenarios, to analyze the impact of various factors on the performance improvements achieved by DABAST. Later in this chapter, we study the performance of DABAST that employs DISCS, and analyze the improvements in QoE metrics achieved by employing DISCS.

7.1 The DABAST architecture

As previously mentioned, DABAST improves video streaming QoE of end users, by reducing the bottleneck of the RAN. This is achieved by employing the CSVD and DISCS algorithms. In this section, we provide a brief reminder of the employed CSVD and DISCS algorithms. Afterwards, we describe the DABAST architecture and how it is implemented in the cellular network.

7.1.1 The CSVD and DISCS algorithms

CSVD and DISCS provide BS-assisted content caching and D2D communication for P2P video content distribution in cellular networks. When we use the term P2P transmission here, we refer to direct transmission of video contents between UEs in the cell over D2D links. With both algorithms, the cell is divided into clusters. To do that, the coverage area of the cell is divided into non-overlapping subareas by the BS. Each one of these subareas

will be a cluster. The BS assigns UEs to clusters based on their locations, and it selects the UEs in the central area of each cluster as SMs of that cluster. SMs are UEs that are used as helpers in the cluster. Only the UEs in the middle of each cluster are selected as SMs, in order to prevent inter-cluster as well as inter-cell interference when the SMs transmit to other UEs in the same cluster using D2D links. After clustering, when a UE requests a video, the BS processes the request and responds as follows:

- Send With Assistance (SWA): if the video file (or parts of it) is available in any of the SMs, the BS will ask the SMs to send the pieces to the requesting UE over D2D links.
- Send To an SM (STSM): if the requested file is not available in the distributed cache (or more copies need to be cached in the cluster) and the requesting UE is an SM, the BS will send the file to that UE over a cellular link, and it will ask the UE to cache the file. This case allows the SMs to cache video files. These files will be available for UEs in the cluster when requested later.
- Distribute To SMs (DTSMs): this case is only used in DISCS. In this case, if a requested video is popular and it is not available in the distributed cache of the cluster, the BS will distribute the segments of the video among the SMs. The BS asks the SMs to cache the pieces (as the video is popular) and forward the received pieces to the requesting UE.
- Send To a UE (STUE): otherwise, the BS will send the file directly to the requesting UE over a cellular link.

7.1.2 Operation of DABAST

Figure 7.1 illustrates the main structure for DABAST. At the bottom, we have the LTE-A network that provides the infrastructure for communication between the BS and UEs over cellular links, and the communication between UEs over D2D links where the UEs exchange data directly without going through the BS. The BS-assisted video caching and P2P/D2D communication protocol (CSVD or DISCS) is implemented on top of that which uses both cellular and D2D communication. DASH-based video streaming takes place on top of these layers, as the transmission of video segments is implemented as per the video content caching and distribution protocol at the layer below.

Figure 7.2 illustrates how DABAST can be implemented in cellular networks. A CSVD server/proxy is used in the RAN at the BS. This provides the processing and networking capabilities needed to implement the cached and segmented video download algorithms. This server can also be used to provide caching capabilities to store popular video files.

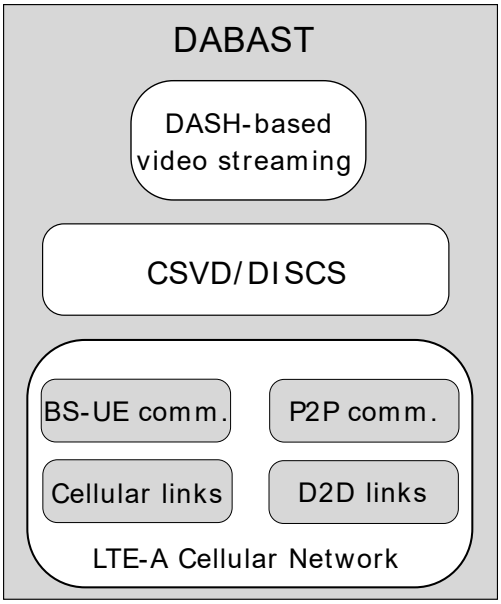

Figure 7.1. The DABAST architecture.

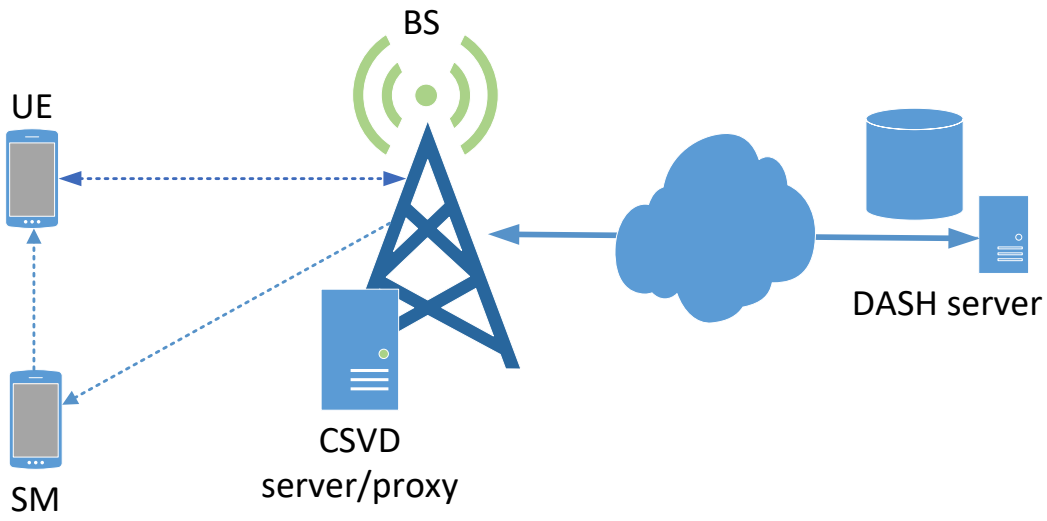

Figure 7.2. Illustration of DABAST implementation.

Streaming clients at the UEs send their requests asking for video segments. These requests are intercepted and processed by the CSVD server. Based on the employed algorithm (CSVD or DISCS), the server decides whether to send the segment from the distributed cache over the D2D channel or get it from the content server and send it over the cellular channel. If the video segment is to be delivered from the distributed cache, an assistance request will be sent to an SM. Otherwise, the request will be forwarded to the DASH server.

Under high traffic load, the BS sends a video segment from the distributed cache (when found) even if the video bit rate of the cached segment does not match the video bit rate requested by the UE. This is to maximize the exploitation of the distributed cache and the D2D channel. Another feature of DABAST is that it can operate in proactive mode. In this mode, DABAST can send up to a certain number of video segments to the user when found in the distributed cache before the segments are requested. This reduces the signaling and latency between the BS and the UE and speeds up transmission of video

segments. However, If the video segment is not available in the distributed cache, the request will be awaited.

Regarding the DASH-controller, we use the buffer-based approach proposed in [41]. This is because in our architecture, the UE could receive a video segment from the BS or from any SM in the cluster. As such, it would be difficult to estimate the data rate at which the next segment will be received.

7.2 Performance evaluation of DABAST

In this section, we study the performance of DABAST with CSVD. Unless the employed algorithm is explicitly mentioned, the name DABAST implies that CSVD is employed. We updated the DEVS model in the previous chapter to model an LTE-A network that implements DABAST. The BS and UE models were updated to operate as per the CSVD algorithm. We executed simulations to evaluate the performance of DABAST in terms of the QoE metrics used in the previous chapter. Table 7.1 shows the simulation setup.

First, we present the results for the performance of DABAST under a single simulation setup and provide a thorough analysis of the results. Then, we run various simulations to study the impact of many parameters on DABAST such as files' popularity, the number of UEs in the cell, etc.

The simulations consider a single LTE-A cell. The urban macro propagation model [86] was used for cellular links with a DL operating carrier frequency of 900 MHz, and a transmission bandwidth of 10 MHz. The D2D channel model at 24 GHz was used for D2D transmission [87].

In each iteration of the simulation, the UEs are uniformly distributed throughout the cell. Clustering takes place in the beginning in case of DABAST where the cell is divided into 9 clusters. The UEs then start requesting video streams.

Table 7.1. Simulation setup.

Parameter	Value
Cellular channel BW (MHz)	10
Cell range (m)	500
Number of clusters	9
BS antenna gain (dB)	12
BS transmission power (dBm)	43
UE antenna gain (dB)	0
UE transmission power (dBm)	21
Noise spectral density (dBm)	-174
Antenna height (m)	15
Transmission model	UTRA-FDD
DL carrier frequency (MHz)	900
Number of requests by a UE	2
Area configuration	Urban
D2D channel BW (MHz)	60
D2D carrier frequency (GHz)	24
D2D transmitter TX power (dBm)	23
D2D large-scale fading std deviation (dB)	4.3
D2D receiver noise figure (dB)	9
D2D TX/RX height from ground (m)	1.5
Segment length (second)	10
Number of buffered segments to start playout	4
Video bit rate levels (kbps)	384, 768, 2000, 4000
Videos length (second)	441

During each iteration of the simulation, each UE requests two video streams. A UE requests a video stream, and after finishing video playout, it requests a second video. The

arrival of requests is generated according a Poisson arrival process. The popularity of videos is generated according to a Zipf distribution to simulate the variable popularity of the videos. The length of the videos is 441 seconds, which is the mean length of a YouTube video [93]. Four video bit rate levels were used as shown in Table 7.1. These are adapted from the H.264/AVC video coding standard [95]. The number of video requests made by each UE in a simulation run is 2.

Regarding the DASH controller, the buffer-based approach in [41] was employed. This is because in our architecture, the UE could receive a video segment from the BS or from any SM in the cluster. As such, it would be difficult to estimate the bit rate at which the next segment will be received. The adaptation algorithm used is a piecewise function, *f(L)*, that uses the length of the playout buffer, *L*, to determine the video bit rate.

Table 7.2. Playout buffer length to video rate mapping.

Playout buffer length (s)	Video bit rate (kbps)
$0 \leq L \leq 90$	384
$90 < L \leq 150$	768
$150 < L \leq 200$	2000
$200 < L$	4000

We assume that the video buffer is long enough to accommodate all received segments. The playout buffer to video rate mapping is shown in Table 7.2. We measured the number of rebufferings, video continuity index, initial delay, and video bit rate levels of the received video segments. Table 7.3 shows the mean values for these measurements. The results in Table 7.3 are for 500 UEs in the cell, Zipf exponent of 1.5, and 500 videos. The average value for each simulation run was calculated. The values below show the mean of all the average values from 50 simulation runs. In addition to the mean, we show

the MoE for 95% confidence interval.

Table 7.3 shows that DABAST achieves improvements over conventional DASH in terms of all the measured metrics above. Regarding the average number of rebufferings, DABAST achieved 50% decrease in the average number of rebufferings, which is a significant improvement. The continuity index is also improved with DABAST due to decreasing the average number of rebufferings as well as the rebuffering time. It is worth mentioning that the average initial delay for conventional DASH is high because in this scenario, there are 500 UEs in the cell requesting video streams and sharing fixed cellular frequency resources (10 MHz). This is also because video playout starts after receiving 4 video segments.

Table 7.3. Simulation results.

	Conventional DASH		**DABAST**	
	Mean	MoE	Mean	MoE
Rebufferings	3.4448	0.0179	1.7272	0.0164
Cont. index	0.7447	0.0009	0.8699	0.0001
Initial delay(sec)	56.881	0.2200	28.654	0.4821
Video bit rate (kbps)	397.27	0.3267	430.16	1.3694

Table 7.3 shows that DABAST also achieves a 50% decrease in the average initial delay, which is also a significant improvement. In addition to the improvements above, DABAST also achieved an improvement in terms of the average video bit rate. Due to the increase in the transmission rates achieved by DABAST, video segments are delivered to UEs much faster than in the case of conventional DASH. This is because in the case of DABAST, the CSVD algorithm is employed, where video segments are sent to many UEs from both the BS (over cellular links) and SMs (over D2D links) as

opposed to only from the BS. This reduces the transmission delay of the first 4 segments needed to start playout, and consequently, reduces initial delay. This also reduces the possibility of video buffer stalling, and hence, reduces the number of rebufferings. There is only a small improvement achieved by DABAST in terms of average video bit rate. This is due to two reasons. First, with both conventional DASH and DABAST, the DASH controller resorts to choosing a lower video bit rate level to increase the video playout buffer length and reduce the number of rebufferings. This means that video bit rate is the last metric that is improved when transmission rate improves. As such, the improvement in terms of the number of rebufferings is usually achieved on the expense of video bit rate. Second, as mentioned in the previous section, under high traffic load, the BS will send a video segment from the distributed cache (when found) even if the segment found in the distributed cache does not match the video bit rate requested by the UE. This is to increase the utilization of the available D2D channel and to speed up the transmission of video segments, as sending from the distributed cache is faster. This means that although DABAST might increase the transmission rate and playout buffer length for some clients (which increases the requested video rate), such users might still receive segments with low video bit rate (from the distributed cache).

Figure 7.3 shows the histogram of the number of rebufferings for both conventional DASH and DABAST. The figure shows that over 96% of the streaming requests have 3 or 4 rebufferings in the case of conventional DASH. With DABAST, on the other hand, half of the streaming requests have 0 rebufferings, and slightly less than half of the streams have 3 or 4 rebufferings. After videos accumulate in the clusters' caches, many video segments will be delivered from the distributed cache in the cell. These segments

will be transmitted faster to the requesting UEs. Moreover, as many of the segments will be sent over D2D links, there will be more cellular resources available for segments that are transmitted over cellular resources, which also reduces the transmission delay for such segments and reduces the possibility of playout interruption, decreasing the number of rebufferings by 50%.

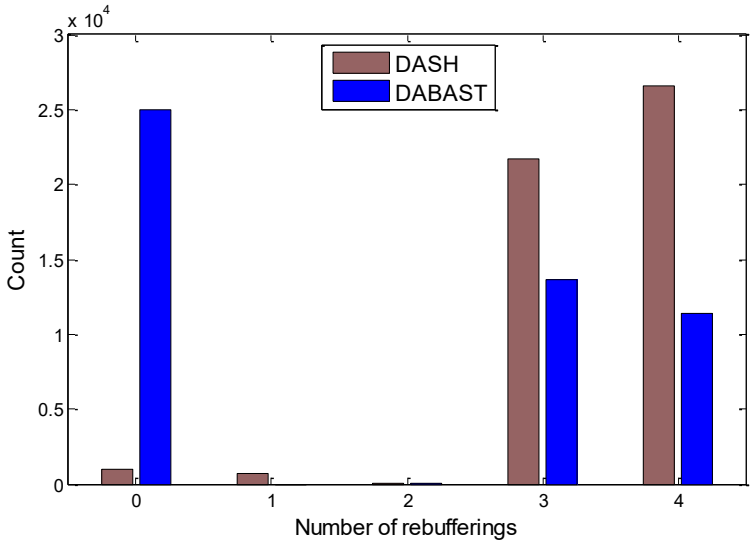

Figure 7.3. Histogram of the number of rebufferings for conventional DASH and DABAST.

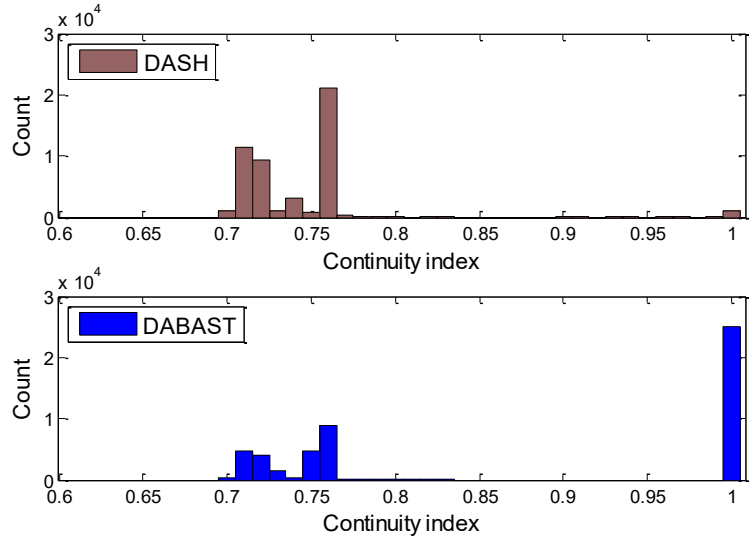

Figure 7.4. Histogram of the continuity-index for conventional DASH and DABAST.

Figure 7.4 shows the histogram of the continuity index for both conventional DASH and

DABAST. The continuity index results match these in Figure 7.3 for the number of rebufferings. Half of the requests with DABAST have a continuity index of 1, which corresponds to zero rebufferings. Less than half of the video streams have a continuity index less than 0.76 with DABAST (corresponds to 3 and 4 rebufferings). However, in the case of conventional DASH, over 96% of the video streams have a continuity index less than 0.76.

Figure 7.5 shows the ECDF of the initial delay for both conventional DASH and DABAST. We can see that the ECDF of DABAST is always higher than that of conventional DASH. For example, the probability of having a stream with initial delay of 20 seconds or less is 0.46 with DABAST, and only 0.09 with conventional DASH.

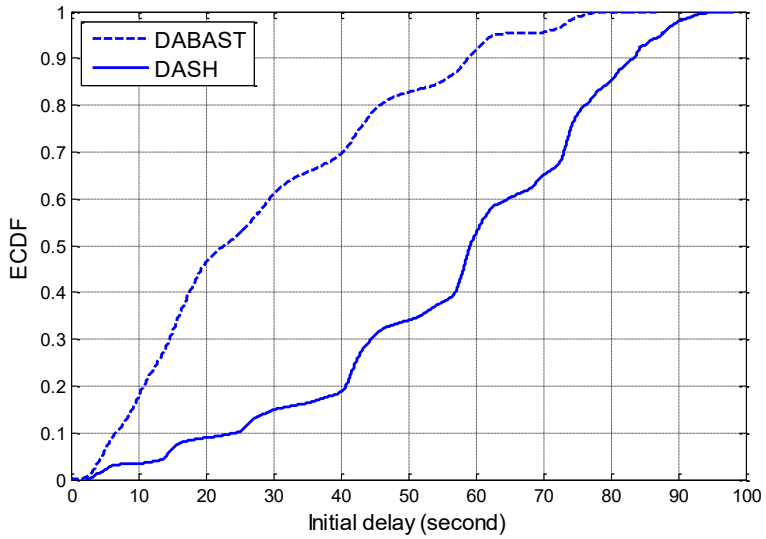

Figure 7.5. ECDF of the initial delay for conventional DASH and DABAST.

Figure 7.5 also shows that 50% of the streams have initial delay of 22.76 seconds or less with DABAST while 50% of the streams have 59.21 seconds or less with conventional DASH. As previously mentioned, with conventional DASH, all the UEs in the cell share the fixed frequency resources (10 MHz cellular channel), while with DABAST, the D2D

channel is exploited for P2P communication in addition to cellular resources. The transmission of video segments from the distributed cache in the cell speeds up the delivery of video segments and significantly reduces initial delay.

Table 7.4 shows the count for the received video segments with each video bit rate level, for both conventional DASH and DABAST. The results show that with DABAST, fewer video segments with 384 kbps were received and more video segments with higher levels (768, 2000, and 4000 kbps) were received. This explains why the average video bit rate (Table 7.3) for DABAST is higher than that for conventional DASH. With DABAST, video segments are delivered faster to the requesting UEs as explained above. As such, clients will have more video segments in the playout buffer, i.e., higher playout buffer length. Consequently, the requested video bit rate will be higher in the case of DABAST.

The results presented in this section show that DABAST provides improvements over conventional DASH in terms of all the measured metrics, which significantly improves the QoE of video streaming in cellular networks. In the following, we investigate the impact of different parameters on the performance of DABAST.

Table 7.4. Count of the received segments with each video bit rate.

	Count	
Video bit rate (kbps)	Conventional DASH	DABAST
384	2207508	2077111
768	31494	149365
2000	10998	19273
4000	0	4251

Figure 7.6 shows the average number of rebufferings for both conventional DASH and DABAST versus the number of UEs in the cell. Figure 7.6 shows that at 300 UEs, the average number of rebufferings is 0 for both conventional DASH and DABAST. This means that the cellular resources are enough with conventional DASH to avoid rebufferings for all the video streams. As the number of UEs increases, the available cellular resources will be shared by more UEs, which reduces the average data rate and increases the transmission delay of video segments. This increases the possibility of video buffer depletion and increases the average number of rebufferings.

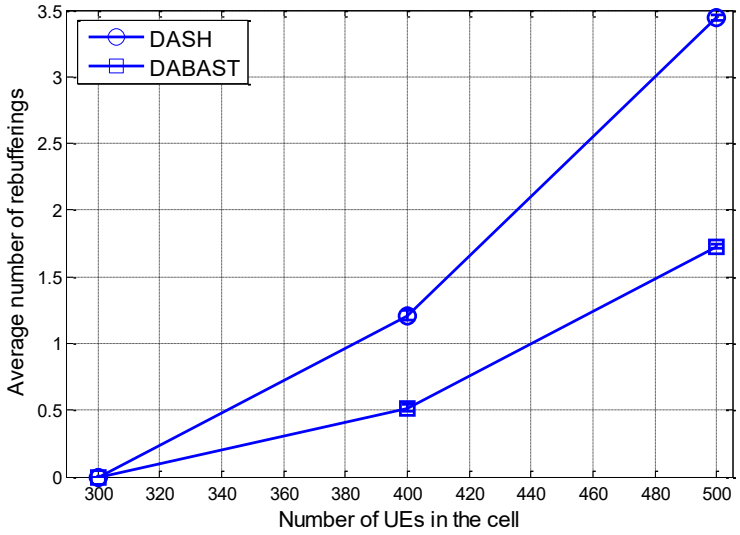

Figure 7.6. Average number of rebufferings versus number of UEs in the cell. Zipf exponent = 1.5 and 500 videos.

Even with DABAST, the average number of rebufferings increases with increasing the number of UEs. This is because we study the case of progressive caching, where in the beginning there are no videos cached, and videos are cached as requested. Figure 7.6 shows that the improvement achieved by DABAST increases by increasing the number of UEs in the cell. Increasing the number of UEs increases the number of requests and also increases the number of SMs in each cluster. This increases the number of cached videos

in a cluster and the percentage of requests that would be satisfied from the cluster's cache, increasing the improvement achieved by DABAST over conventional DASH.

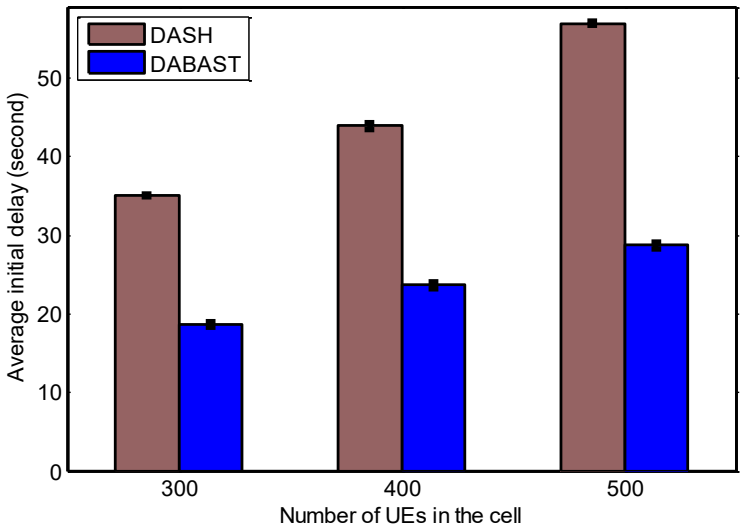

Figure 7.7. Average initial delay versus number of UEs in the cell. Zipf exponent = 1.5 and 500 videos

Figure 7.7 shows the average initial delay for both conventional DASH and DABAST versus the number of UEs in the cell. Figure 7.7 shows that although the average number of rebufferings is zero at 300 UEs for both approaches (as in Figure 7.6) DABAST still achieves improvement in terms of the average initial delay at 300 UEs. Furthermore, it can be noticed that the gain achieved by DABAST over conventional DASH increases with the number of UEs. This is for the same reason explained above for the average number of rebufferings.

As mentioned above, the Zipf distribution was used to model the popularity of videos. The Zipf distribution has one parameter, namely the Zipf exponent. This exponent controls the relative popularity of the videos. When the value of the Zipf exponent increases, the relative popularity of some videos in the list increases. This increases the possibility of requesting these videos.

Figure 7.8 shows the average number of rebufferings versus the Zipf exponent. As can be seen, the average number of rebufferings decreases as the Zipf exponent increases. As the popularity of some videos increases, higher percentage of the requests will be found in the distributed cache and delivered over the D2D channel (rather than the cellular channel). This speeds up the transmission of many segments and decreases the possibility of playout buffer depletion, which decreases the average number of rebufferings.

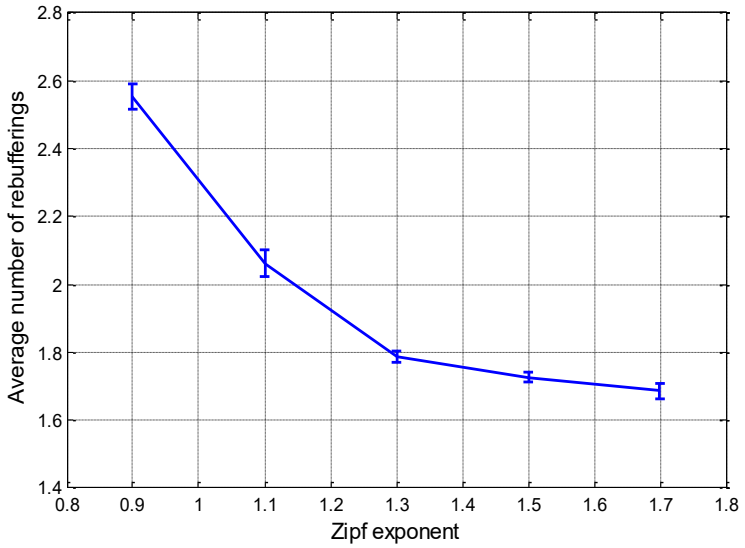

Figure 7.8. Average number of rebufferings versus the Zipf exponent. 500 UEs and 500 videos.

Figure 7.8 shows that the improvement achieved with increasing the Zipf exponent eventually slows down. This is because in our scenario, each UE requests only two videos. Increasing the number of requests made by each UE further increases the exploitation of cached contents and increases the improvement achieved by DABAST. This is because cached videos will be further used by the later requests.

Figure 7.9 shows the average number of rebufferings for DABAST versus the number of videos available to request from. As can be seen, there is no considerable effect for the number of videos on the average number of rebufferings. As per the Zipf distribution,

having more videos will not cause a noticeable impact on the probability of requesting the popular files. This means that for a certain Zipf exponent, although the number of videos increases, cached contents will still be exploited as long as there are popular videos.

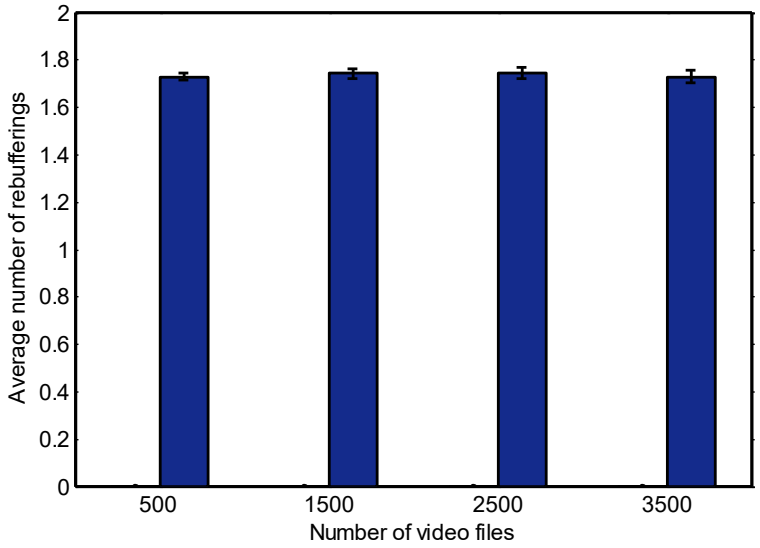

Figure 7.9. Average number of rebufferings versus the number of videos. 500 UEs and Zipf exponent = 1.5.

It is worth mentioning that DABAST should be even more beneficial when considering background traffic. To get a feel of the improvement achieved by DABAST in the case background traffic is present, we ran simulations with 300 UEs in the cell and with the same setup in Table 7.1. Unlike the previous simulations above, we consider that background traffic is present in the cell, and that the BS dedicates 5 MHz of the channel for background traffic and 5 MHz for video streaming traffic. Let us recall that in the case background traffic is absent and the whole channel is available for video streaming traffic (300 UEs), users experienced 0 rebuffering with both DASH and DABAST. On the other hand, in the case background traffic is present, the results have shown that with conventional DASH, the average number of rebufferings is 4.47, while with DABAST, the average number of rebufferings is 2.35. This shows that DABAST in this case has

achieved 47.4% reduction in the average number of rebufferings. This shows how DABAST is even more beneficial and can achieve further gains in the case background traffic is present.

7.3 The impact of the caching and distribution algorithm on QoE

In Chapter 5, we have studied the performance of CSVD and DISCS in terms of data rates. Results have shown that DISCS achieves significant improvement over CSVD in terms of the average data rate. Furthermore, in the previous section, we have studied the performance improvement achieved by DABAST that employs CSVD (DABAST-CSVD) in terms of video streaming QoE. Results have also shown that significant improvements can be achieved using DABAST-CSVD in terms of all the measured QoE metrics. In this section, we study how much improvement DABAST can further achieve if DISCS is employed instead of CSVD, i.e., we want to study if the significant improvement achieved by DISCS over CSVD in terms of data rate translates into significant improvement in terms of QoE.

Remember that with DISCS, in addition to the 3 cases considered in CSVD, that are listed in 7.1.1 (and discussed in detail in 3.1), one more case is employed (DTSMs). In this case, if a requested video is popular (requested *n* times) and it is not available in the distributed cache of the cluster, the BS will distribute the pieces among the SMs in the cluster. The BS asks the SMs to cache the received pieces (as the file is popular) and forward them to the requesting UE. As previously mentioned, this case helps speeding up accumulation of popular video files in the distributed cache of the cluster. It also allows for more parallelism and load balancing among SMs when sending video files from the

distributed cache of the cluster. This should increase the utilization of the D2D channel and speeds up the transmission, and consequently increase the average data rate.

We executed simulations to evaluate the performance of DABAST that employs DISCS (DABAST-DISCS) in terms of the same QoE metrics. The simulation setup in the previous section (Table 7.1 and Table 7.2) was also used here. Table 7.5 shows the mean values along with the MoE values for 95% confidence interval for the measured QoE metrics. The table shows the values for DABAST-CSVD and DABAST-DISCS.

Table 7.5. Simulation results (DABAST-CSVD vs. DABAST-CSVD).

	DABAST-CSVD		**DABAST-DISCS**	
	Mean	MoE	Mean	MoE
Rebufferings	1.7272	0.0164	1.6294	0.0169
Cont. index	0.8699	0.0001	0.8763	0.0011
Initial delay(sec)	28.654	0.4821	21.192	0.4163
Video bit rate (kbps)	430.16	1.3694	448.57	0.9607

The results in Table 7.5 are for 500 UEs in the cell, Zipf exponent of 1.5, and 500 videos. As in the previous section, the average value for each simulation run was calculated. The values below show the mean of all the average values from 50 simulation runs. From Table 7.5, we can see that as expected, DABAST-DISCS provides improvement over DABAST-CSVD in terms of the measured QoE metrics. Table 7.5 shows that DABAST-DISCS reduced the average number of rebufferings from 1.73 to 1.63, which increased the continuity index. With DABAST-DISCS, the initial delay is also reduced from 28.7 seconds to 21.2 seconds, which is a significant improvement. Furthermore, the average video bit rate increased by 18 kbps with DABAST-DISCS.

Although the results above show that DABAST-DISCS improves the QoE metrics, one can see that the only significant improvement achieved by DABAST-DISCS is in terms of the initial delay. Only slight improvement is achieved in terms of the average number of rebufferings and continuity index. One would expect higher gains by DABAST-DISCS over DABAST-CSVD given the results in Chapter 5 which show that DISCS achieves roughly double the average data rates in a similar scenario (transient phase results). In the following, we present further analysis of the above results to understand this behavior.

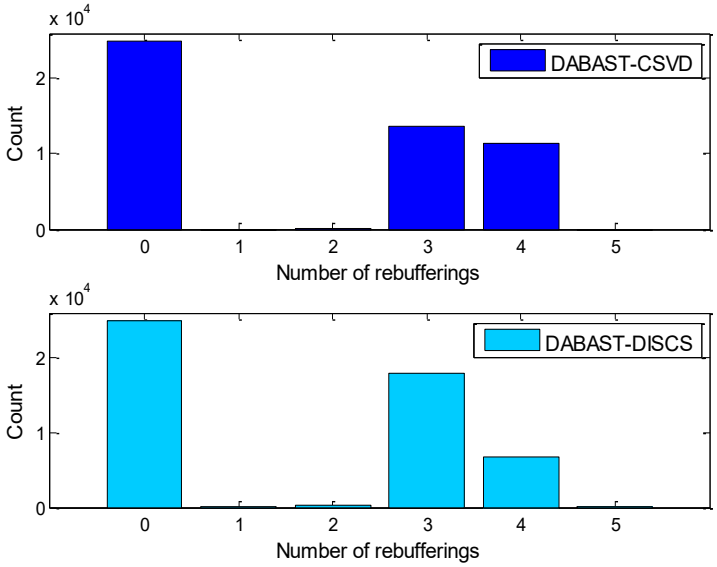

Figure 7.10. Histogram of the number of rebufferings for DABAST-CSVD and DABAST-DISCS.

Figure 7.10 depicts the histogram of the number of rebufferings for DABAST-CSVD and DABAST-DISCS. From the figure, one can see that the main difference is that with DABAST-DISCS, higher number of video streams have 3 rebuffering and fewer number of video streams have 4 rebufferings. This explains why the average number of rebufferings with DABAST-DISCS is less than that with DABAST-CSVD.

To provide a more quantitative evaluation of the difference, we show the relative frequency histogram for DABAST-CSVD and DABAST-DISCS in Figure 7.11. The

figure shows that with both DABAST-CSVD and DABAST-DISCS, 50% of the video streams have 0 rebufferings. With DABAST-CSVD, 27.3% of the video streams have 3 rebufferings, and 22.7% of the streams have 4 rebufferings. With DABAST-DISCS, 36.0% of the video streams have 3 rebufferings, and 13.3% of the streams have 4 rebufferings. This improvement in the number of rebufferings is expected, as DABAST speeds up video caching, and improves the rate at which video segments are delivered to requesting UEs.

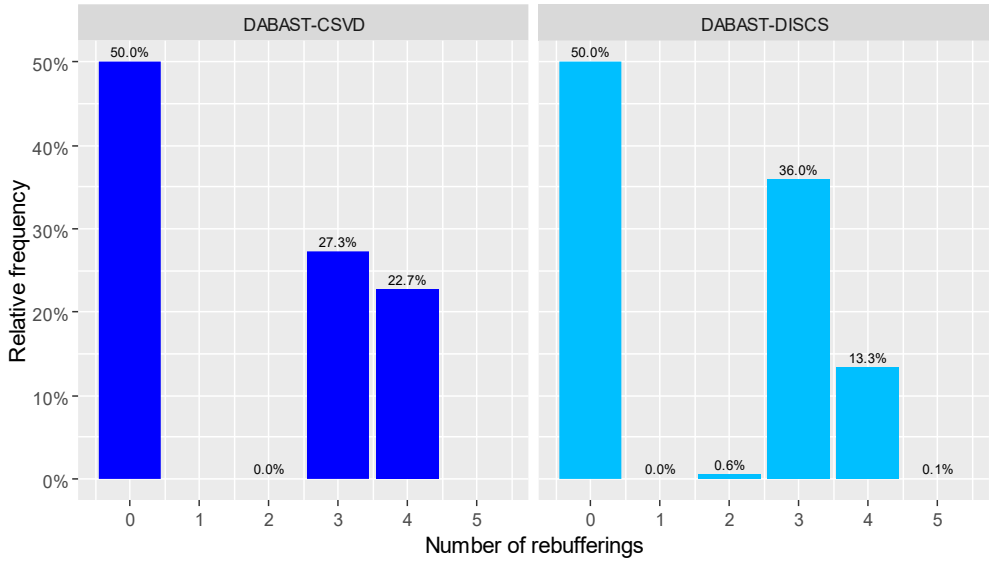

Figure 7.11. Relative frequency histogram of the number of rebufferings for DABAST-CSVD and DABAST-DISCS.

Figure 7.11 also shows that with DABAST-DISCS, a very small percentage of the video streams (0.1%) have 5 rebufferings. This increase in the number of rebufferings experienced by a slight percentage of the streams is a result of the DTSMs case, where video segments are sent to the requesting UEs in two steps, i.e., the segment is sent to the SM first, and then sent to the UE by the SM. Despite of this increase to a very small percentage of the UEs, DABAST-DISCS still achieves lower initial delays and number of rebufferings, on average. This means that the DTSMs case is beneficial to the cell, as expected.

Although the results show that DABAST-DISCS decreases the number of rebufferings, one can argue that DABAST-DISCS is expected to achieve higher gains in terms of the number of rebufferings, as it significantly improves the average data rate. To further investigate the results and explain this behavior, we separate the results for each request.

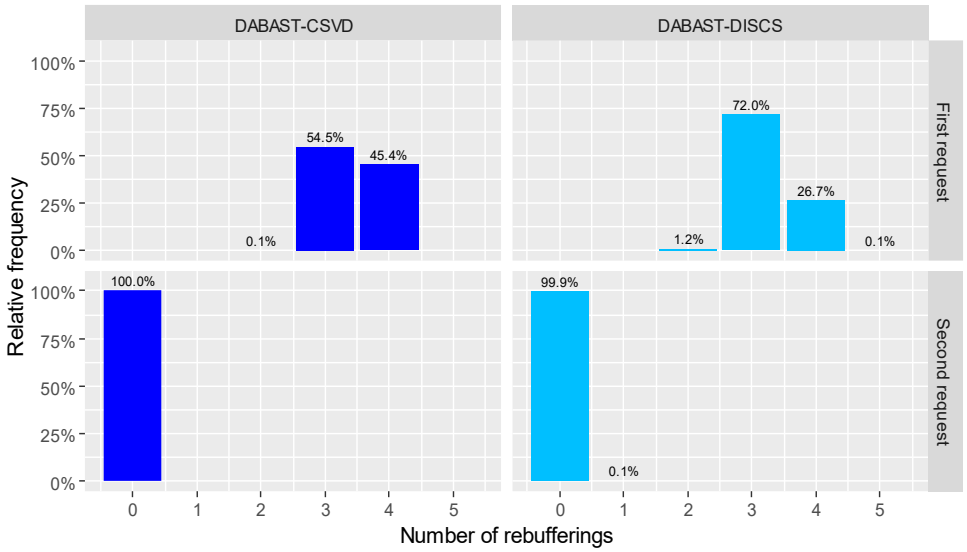

Figure 7.12. Relative frequency histogram of the number of rebufferings in each request for DABAST-CSVD and DABAST-DISCS.

As previously mentioned, each UE in the simulations makes 2 video streaming requests. After playout of the first video, a UE would stay idle for a random period of time, and then generates another request for a video stream. Figure 7.12 shows the relative histogram for the number of rebufferings of each request for both DABAST-CSVD and DABAST-DISCS. As with the previous figure, the histograms on the right side are for DABAST-CSVD, and the ones on the left are for DABAST-DISCS. However, in this figure, the histograms on the top are for the first requests, while the ones at the bottom are for the second requests. Figure 7.12 shows that for DABAST-CSVD, all the rebufferings take place during the first set of video streams. All the second video streams have 0 rebufferings. This is because at the time most of the first video streams start, there

are no video segments cached in the distributed caches. Hence, most of the video segments will be delivered over the cellular channel. As such, the limited cellular channel will be shared by the large number of users, which means the average data rate at which these segments are delivered is low and explains the high number of rebufferings.

The figure shows that with DABAST-CSVD, all the second set of streams have 0 rebufferings. By the time the second set of streams starts, there will be many video segments cached in the clusters. Hence, many of the segments will be delivered over D2D links, which eliminates rebufferings for those streams. This also relaxes the bottleneck of the RAN because, at this time, only a portion of the video segments will be sent over the cellular channel. Because the cellular channel is now shared by a much lower number of UEs, the data rates will increase, and rebufferings will be avoided for these video streams as well.

With DABAST-DISCS, almost all rebufferings occur in the first set of video streams, as 99.9% of the second set of video streams have 0 rebufferings, and only 0.1% of the second set of video streams have 1 rebufferings. This is for the same reason all the rebufferings with DABAST-CSVD occur during the first set of video streams. Initially, there are no video segments available in the distributed cache, and hence, all video streams will experience multiple rebufferings. By the time the second set of streams starts, there will be many video segments cached in the clusters. Hence, many of the segments will delivered over D2D links, which eliminates rebufferings for those streams, and relaxes the RAN bottleneck for segments delivered over cellular resources. The one difference here is that there is a very small portion of the second set of video streams that

still get its segments with the DTSMs case, which causes 1 rebuffering to 0.1% for the second set of video streams.

The results in Figure 7.12 also explain why DABAST-DISCS does not achieve very significant improvement in terms of the average number of rebufferings over DABAST-CSVD. As discussed in Chapter 5, DISCS achieves significant improvement in terms of the average data rate over CSVD. This is because in the case of DISCS, many files will be sent in parallel from multiple SMs (as opposed to one SM). This causes further parallelism in sending video segments and better load balancing between SMs, which speeds up the transmission of video files and increases the average data rate. However, we have seen from Figure 7.12 that video streams with segments delivered over D2D links already have 0 rebufferings, even in the case of DABAST-CSVD. As such, the increase in the average data rate achieved by DABAST-DISCS will not translate into reduction in the average number of rebufferings, as all rebufferings take place in the first set of streams when video segments are not cached. This means that the improvement in the average number of rebufferings gained by DABAST-DISCS over DABAST-CSVD is due to the fact that DISCS speeds up video caching and achieves better hit ratio as discussed in the previous chapter, which increases the percentage of requests that are satisfied from the cluster's cache and speeds up the relaxation of the RAN bottleneck.

Figure 7.13 shows the histogram of the continuity-index for both DABAST-CSVD and DABAST-DISCS. As expected, 50% of the streams have a continuity index of 1. With DABAST-DISCS there is more concentration of the values around 0.76 and less concentration of values between 0.71 and 0.75. This agrees with the results for the

number of rebufferings. However, with DABAST-DISCS, there are few continuity index values less than 0.7. These values correspond to streams with 5 rebufferings.

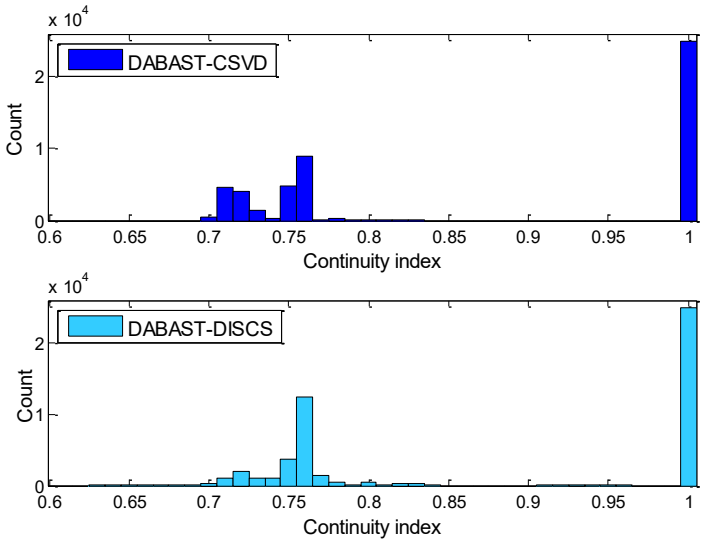

Figure 7.13. Histogram of the continuity-index for DABAST-CSVD and DABAST-DISCS.

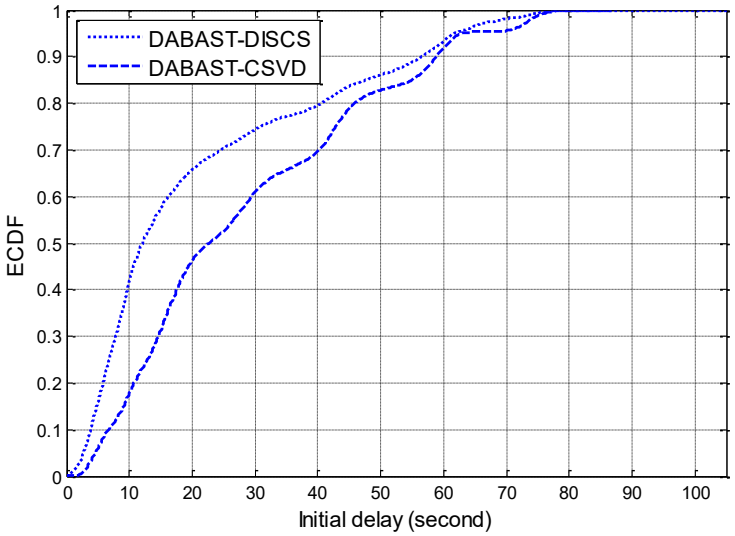

Figure 7.14. ECDF of the initial delay for DABAST-CSVD and DABAST-DISCS.

Figure 7.14 shows the ECDF of the initial delay for DABAST-CSVD and DABAST-DISCS. From the figure, we can see that the ECDF of DABAST-DISCS is always higher than that of DABAST-CSVD, until they meet at 77 seconds, at ECDF value that is very close to 1 (0.997). This means that up until 77 seconds, it is always more probable to

achieve less initial delay with DABAST-DISCS. For example, the probability of having a stream with initial delay of 20 seconds or less is 0.66 with DABAST-DISCS, and only 0.46 with DABAST-CSVD. However, after 77 seconds, the ECDF values of DABAST-CSVD starts exceeding that of DABAST-DISCS. The ECDF values of DABAST-CSVD reaches 1 at 86.7 second, while with DABAST-DISCS it does at 105.4 second. This means that with DABAST-DISCS, a very low percentage of the UEs might suffer from higher initial delays. These are UEs that obtain their video segments through the DTSMs case, where the video segments are sent in two hops, i.e., each video segment is sent to a SM first, and then the SM send it to the requesting UE. However, as can be seen from Figure 7.14, this is an extremely small percentage of the UEs that suffer from additional delay and employing the DTSMs case is ultimately very beneficial for the cell, as the average initial delay is significantly lower with DABAST-DISCS.

Regarding the average video bit rate, we can also see that DABAST-DISCS did not achieve a significant improvement over DABAST-CSVD (only 4.3% improvement). This can be explained as follows. As most of the cached video segments are downloaded during high traffic load. These segments are usually downloaded with low video bit rate. By the time the second set of video streams starts, there will be many segments available in the clusters' caches. However, most of these segments have low video bit rate. As DABAST, under high traffic load, sends available segments from the clusters' caches over D2D links (in the case an SM is available), most of the segments transmitted over the D2D channel will be sent from the distributed cache with low video bit rate. Although these segments are transmitted with higher data rates in the case of DABAST-DISCS, this does not increase the video bit rate for these segments. DABAST operates in this fashion to

save the valuable cellular resources and exploit them for sending video segments that are not available in the clusters' caches to avoid rebufferings as much as possible.

A cached video segment is sent over the cellular channel only when there are no SMs available to send the segment. These video segments that are sent over cellular resources (despite being cached) will usually be sent with high video bit rate as such segments are usually requested with high video bit rate. This is because streams with cached video segments usually have long playout buffer length, as most of their segments are sent over the D2D channel with higher data rates. Furthermore, such video streams have even longer playout buffer length in the case of DABAST-DISCS, as video segments are sent with higher data rate than these with DABAST-CSVD. Because of that, video segments that are sent over cellular resources (despite being cached) will be sent with higher video bit rate in the case of DABAST-DISCS when compared to DABAST-CSVD. This explain the small improvement achieved by DABAST-DISCS over DABAST-CSVD in terms of the average video bit rate.

This behavior, explained above, can be seen in Table 7.6, which shows the number of video segments received with each video bit rate, for DABAST-CSVD and DABAST-DISCS.

Table 7.6. Count of the received segments with each video bit rate.

	Count	
Video bit rate (kbps)	DABAST-CSVD	DABAST-DISCS
384	2077111	2089649
768	149365	105874
2000	19273	46179
4000	4251	8298

Table 7.6 shows that in the case of DABAST-DISCS, fewer segments are received with video bit rate of 768 kbps, and more segments (about the double) are received with 2Mbps and 4Mbps video bit rates, when compared to DABAST-CSVD. While this is beneficial for the streams that receive video segments with high video bit rates, it increases the RAN bottleneck and decreases the average data rate for other UEs receiving video segments exclusively over cellular resources. In the next chapter, we present an approach for resource management in DABAST that helps avoiding such unfairness in resource allocation.

7.4 Summary

In this chapter, we propose and evaluate an architecture, namely DABAST, that improves the QoE of video streaming in cellular networks with high user density. The proposed architecture employs the CSVD and DISCS algorithms (discussed in Chapter 3 and Chapter 5). The proposed algorithms also employ DASH due to its advantages and to allow support for DASH-based applications.

We use the DEVS formalisms to build a model for the proposed architecture in an LTE-A network. Simulations based on this model are used to evaluate the performance of DABAST. We first consider DABAST which employs CSVD. We provide a thorough analysis of the performance improvements achieved by DABAST in terms of many video streaming QoE metrics. Results show that DABAST achieves clear improvements in terms of all the studied QoE metrics. Furthermore, we investigate the performance of DABAST in many scenarios, to analyze the impact of various factors on the performance improvements achieved by DABAST. Later in this chapter, we study the performance of

DABAST that employs DISCS, and analyze the improvements in QoE metrics achieved by employing DISCS.

Chapter 8

QoE-aware DABAST: Buffer-based cellular resource allocation

With the increasing demand for video streaming applications over cellular networks, video streaming QoE is becoming a main concern for cellular network operators. As users pay for the service, they expect to get good QoE in return. According to [96], 90% of customers switch to another provider before complaining about poor service. This makes it necessary for cellular network operators to employ proactive approaches that monitors the quality as perceived by the user and avoid its degradation, instead of waiting for user's complaint. As such, video streaming QoE needs to be considered in network management and operation.

In the context of DABAST, we have seen in the previous chapter that despite the significant improvements achieved by DABAST, some users did not receive video streaming service with good QoE, due to the repetitive video rebuffering. Furthermore, the results for DABAST-CSVD and DABAST-DISCS have shown that a significant increase in the QoS does not always translate into a significant increase in the end-user QoE due to the different factors involved in video streaming QoE. This is also due to the dynamic attributes of video contents, like video bit rate, that could change during video transmission. For instance, increasing the data rate for a user with a smooth and uninterrupted video playout without increasing the video bit rate will not have a big

impact on the end-user QoE. As such, making DABAST aware of video streaming QoE for users in the cell can further increase the QoE gains achieved by DABAST. QoE awareness can be used in DABSAT to make sure that a minimum QoE level is delivered to all users, given the current traffic load and available network resources. QoE awareness can be also employed to maximize the QoE achieved over the network, or to achieve a balance between both objectives.

QoE awareness can be employed in different aspects of DABAST. The first and most crucial aspect is cellular resource allocation. Using QoE-aware cellular resource allocation, resources can be assigned to users according to some QoE-related metric. For instance, higher priority can be given to users who are about to experience imminent playout buffer depletion to avoid rebuffering and minimize the number of rebufferings experienced by users. Employing such QoE-aware approach is vital in DABAST. This is because under high traffic load, the scarce and valuable cellular resources should be allocated to users who receive their video segments exclusively through the BS, as these users usually experience low video streaming QoE.

Another aspect where QoE awareness can be employed is choosing the video bit rate of cached segments. When the traffic load is high, the first goal would be to avoid video rebufferings. As such, video segments will be downloaded with low video bit rate. However, when the traffic load is lower, and users are not experiencing rebufferings, popular videos can be transmitted with high video bit rate. This should considerably increase the video bit rate of segments transmitted later from the distributed cache, and hence increase video streaming QoE.

The third aspect in which QoE-awarness can be used is SM assignment to requesting UEs. Considering that SMs can accept a certain number of concurrent assists, and that different SMs will have different video segments with different video bit rates, assistant requests should be sent to SMs in a way that maximizes the QoE for the users in the cell. For instance, SMs can be assigned to minimize the number of rebufferings, maximize the achievable video bit rate, or considering both objectives.

We developed QoE-aware DABAST to consider the QoE of users in the cell in each one of the aspects described above. In this chapter, we focus on the first aspect, i.e., cellular resource allocation. In the next chapter, we present our proposed techniques for high rate caching and SM assignment optimization. We evaluate the performance improvements achieved by all these techniques through computer simulations.

8.1 Cellular resource allocation

The downlink scheduler is a crucial component in LTE-A systems due to its importance in efficient radio resource utilization. The downlink scheduler allocates radio resources to the UEs in the cell according to some objective (e.g., maximize the cell's aggregate data rate).

In LTE-A, the radio spectrum is accessed using OFDMA in the DL [97]. With OFDMA, the radio channel is divided in both the time and frequency domains as shown in Figure 8.1. In time, the channel is divided into sub-frames, and each sub-frame is 1ms. Each sub-frame is divided into 2 slots of 0.5 ms each. In Frequency, the channel is divided into sub-channels of 180 kHz each. A unit that is composed of one slot in time (0.5 ms) and one sub-channel in frequency is referred to as one Resource Block (RB). In Figure 8.1,

the small gray square represents one RB. In LTE-A, RB is the smallest schedulable radio resource unit. The number of RBs in the channel depends on the channel bandwidth. For example, a 10 MHz channel contains 50 RBs while a 5 MHz channel contains only 25 RBs.

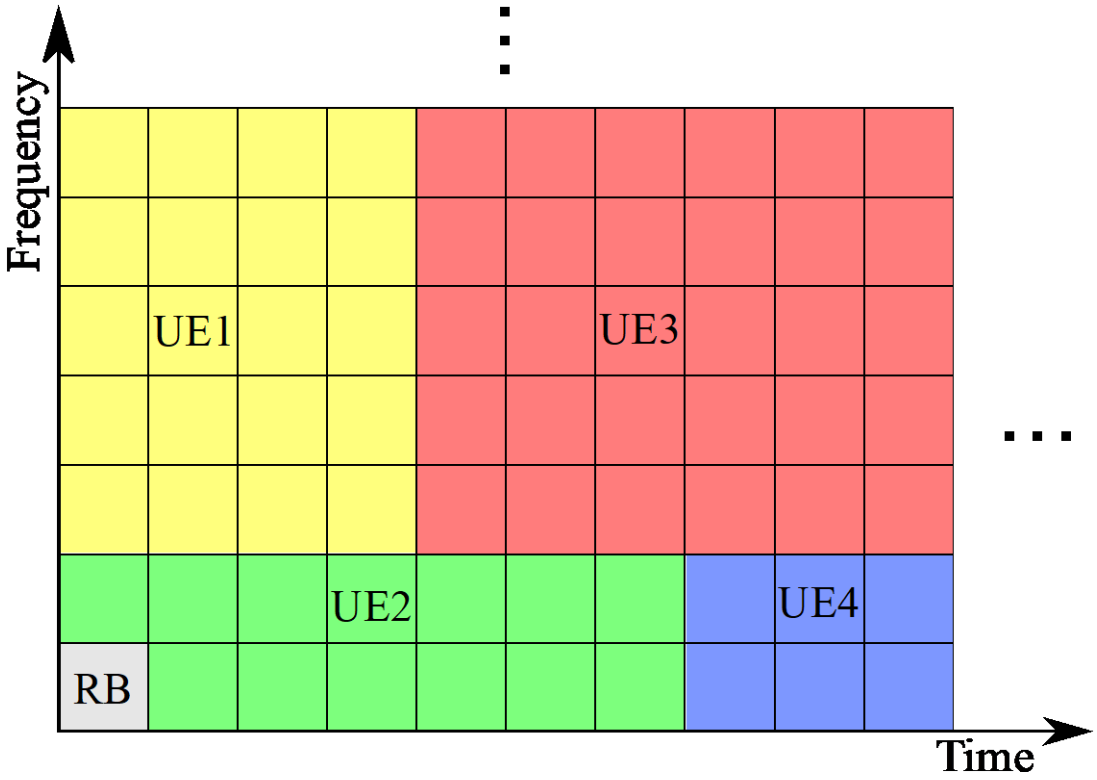

Figure 8.1. Time-frequency resource grid.

In LTE-A, the scheduler is implemented at the BS, and it is responsible for allocating RBs to active UEs. For example, Figure 8.1 shows that the yellow RBs are allocated to UE1, the green RBs are allocated to UE2, and so on. Scheduling takes place every TTI to decide which UE will be allocated RBs in the next 1ms interval, based on a certain scheduling metric.

Many scheduling algorithms have been proposed for LTE-A. Round Robin (RR) is one approach that can be used to allocate frequency resources to UEs. It is a simple approach

as it allocates resources to UEs in a RR fashion. For instance, all the RBs in a TTI can be allocated to a UE each time. The advantages of the RR algorithm are its simplicity and fairness. However, it does not consider the quality of the channel between the UE and the BS, and hence, it does not maximize the system throughput. Furthermore, although RR might be fair in a traditional cellular network, it will not be fair in the case of DABAST because some UEs will get video segments over the D2D channel. If this data transmitted over the D2D channel is not taken into consideration, users who get some video segments over the D2D channel and others who get all their video segments over the cellular channel will be treated equally. One can tell that this, in addition to introducing unfairness, might also increase the delay to transmit segments to users who get video segments exclusively over the cellular channel, and consequently increase the possibility of rebuffering for such users.

Another algorithm that is widely used in LTE-A systems is Proportional Fair (PF) scheduling [98]. PF scheduling tries to achieve a balance between maximizing the system throughput and achieving fairness among the UEs competing for the cellular resources. This can be achieved by considering for each UE both the current instantaneous rate for that UE, as well as the recent average throughput of that UE.

The scheduling algorithms above try to optimize certain QoS objectives. For example, RR scheduling tries to achieve fairness in allocating resources so that the UEs in the cell would experience comparable average data rates. PF scheduling tries to maximize the system throughput by considering the instantaneous data rate of the UEs while being relatively fair in allocating resources at the same time. As such, the algorithms above, based on QoS metrics, do not guarantee that video streaming QoE metrics for users in the

cell will be maximized, or at least, try to improve the situation for users who are having low video streaming QoE. Due to the various metrics involved in video streaming QoE and the dynamic attributes of video contents, scheduling resources for video streaming users is more complicated. Traditional scheduling algorithms that are oblivious to the end-user QoE and to the characteristics of video contents might result in low QoE for high number of users. For instance, if there are two users who are watching two different video streams with different video bit rates, providing the two users with the same data rate might result in many rebufferings for the user with the higher video bit rate. User satisfaction is very important for network operators and need to be considered in network operation. As such, there is a need for QoE-aware scheduling algorithms that take into account video streaming QoE when allocating resources to users.

In addition to the above, the nature of DABAST, where some video segments can be transmitted over the D2D channel, adds one more layer to the complexity of the problem. The cellular resources should be allocated taking into consideration that some UEs will get their segments over the D2D channel, and hence, the cellular resources need to be utilized to help other users who are not fortunate to receive segments over the D2D channel (as their segments are not cached).

In this section, we propose a scheduling algorithm for DABAST that takes into account the playout buffer length in the UEs. The metric is similar to the one proposed in [78] for traditional LTE-A systems. However, our algorithm is updated to consider the reported playout buffer length along with estimation of video contents recently transmitted over both the cellular and D2D channels (from all sources). It takes into account all the information above to allocate the cellular resources in a way that improves the QoE for the

users in the cell. Results show that our algorithm improves the achieved video streaming QoE for users in the cell when compared to RR and state-of-the-art PF scheduling.

In the following subsections, we provide a detailed description of PF scheduling, followed by our Buffer-Based scheduling (BB). Then, we provide simulation results for DABAST with all the scheduling algorithms above.

8.1.1 Proportional fair scheduling

PF scheduling tries to achieve a trade-off between system throughput and fairness among the UEs in the cell [98]. It tries to maximize the system throughput by allocating resources to the UE that achieves the highest instantaneous data rate. At the same time, it tries to maintain fairness among the UEs competing for cellular resources by considering the recent average throughput of each UE.

In LTE-A, the BS regularly receives Channel Quality Indicator (CQI) reports from the UEs in the cell [98]. CQI conveys information to the BS on how good the communication channel quality is between the BS and the sending UE. From the information in the CQI, the BS estimates the supported instantaneous data rate, $\hat{r}_k[n]$, for each user k. The PF scheduler then selects the user k' for transmission that has the maximum scheduling metric [98], as follows,

$$k' = \max_k \left\{ M_k^{PF}[n] \right\} = \max_k \left\{ \frac{\hat{r}_k[n]}{T_k[n]} \right\}, \tag{8.1}$$

where $M_k^{PF}[n]$ is the PF scheduling metric for user k, $T_k[n]$ is the recent average throughput for user k over the past window of N transmission intervals, and n denotes the

current scheduling interval.

The recent average throughput for user *k* is calculated as follows,

$$T_k[n] = (1 - \frac{1}{N})T_k[n-1] + \frac{\lambda_k}{N} r_k[n], \tag{8.2}$$

where, *N* is the length of the window over which the average throughput is calculated (sometimes referred to as the memory of the filter), $r_k[n]$, is the instantaneous data rate at scheduling interval *n*, and $\lambda_k[n]$ is the activity factor, and it equals 1 if user *k* is scheduled for transmission in the *nth* TTI and 0 otherwise.

From the above, we can see that the scheduling favors UEs with relatively higher channel quality by having the estimated instantaneous data rate in the nominator, and at the same time, try to maintain fairness by having the recent average throughput in the denominator.

Considering the channel instantaneous rate is very beneficial when employing PF scheduling. However, in the context of DABAST, the real advantage is using the recent average throughput of the users in addition to the instantaneous rate. As previously discussed, in DABAST, some video segments are transmitted over the cellular channel, while other video segments are transmitted over the D2D channel. By maintaining the recent average throughput of each UE, we consider the video data that is transmitted over both the cellular and the D2D channel. Hence, for UEs that received segments over the D2D channel, the recent average throughput will be relatively high, which consequently leads to favoring UEs with low recent average throughput when allocating cellular resources. Usually, these are UEs that receive their video segments exclusively over the

cellular channel. This should reduce the number of rebufferings experienced by such users and consequently improve their QoE.

8.1.2 Buffer-based scheduling

Although PF scheduling provides advantages over RR scheduling, it still does not take into consideration the end-user QoE. As mentioned before, this could be a limiting factor for DABAST. In this section, we propose a scheduling algorithm, to use with DABAST, that takes into consideration the reported length of the playout buffer at the client side. Similar metrics have been proposed for traditional LTE-A system. However, our BB scheduling algorithm is proposed to operate in DABAST and considers not only the reported playout buffer length, but also an estimation of the recently transmitted video duration over both the cellular and the D2D channels (from all SMs).

In BB scheduling, the UEs signal their playout buffer length to the BS. This update can be sent in certain cases; when the change in playout buffer length from the last reported value exceeds a certain threshold (e.g., 5 seconds), or after a certain period passes from the last update. The update can also be sent when the client status changes (e.g., playing to rebuffering). The main idea is that the BS will allocate resources to the UEs giving more priority to UEs with low playout buffer length to avoid rebufferings.

In our BB algorithm, the BS updates the scheduling metric for each UE every TTI. The scheduling metric considers the following:

- The last reported value of the playout buffer length
- An estimation of the video content length transmitted over the cellular channel since the last update

- A third part that takes into account the segments that are being transmitted by SMs over the D2D channel

To implement the above, the BB scheduler selects the user k' for transmission that has the maximum scheduling metric, as follows,

$$k' = \max_k \left\{ M_k^{BB}[n] \right\}, \tag{8.3}$$

where the BB scheduling metric, $M_k^{BB}[n]$, is calculated as follows,

$$M_k^{BB}[n] = V_{\max}[n] - \left(b_k[n] + p_{k,c}[n] + p_{k,d}[n] \right), \tag{8.4}$$

where $V_{\max}[n]$ is the current maximum video length, which is found as follows,

$$V_{\max}[n] = \max_k \left\{ v_k[n] \right\}, \tag{8.5}$$

where $v_k[n]$ is the length of the current video played by user *k*. $b_k[n]$ is the playout buffer length in seconds for user *k,* as reported in the last update, and $p_{k,c}[n]$ is the recently transmitted playout time over cellular resources since the last update, calculated as follows,

$$p_{k,c}[n] = \frac{d_k[n]}{\zeta_k[n]}, \tag{8.6}$$

where $d_k[n]$ is the amount of video data transmitted to user *k* since the last update, and $\zeta_k[n]$ is the current video bit rate of the segment transmitted to user *k* over the cellular

channel.

$p_{k,d}[n]$ is used to take into account the segments that are being transmitted to user *k* via D2D communication since the last update from the UE, and it is calculated as follows,

$$p_{k,d}[n] = N_d \times w \times L \times \sigma_k, \tag{8.7}$$

where N_d is the number of segments currently transmitted over the D2D channel and *w* is the weight given for these segments, i.e., the ones the BS sent *Assistance Request* message for, but they are not received yet. *L* is the length of the segment in seconds. Since such segments are being transmitted now, they should be taken into account when allocating cellular resources. This allows the scheduler to early distinguish UEs that are being sent video segments over the D2D channel. As we will see, this is also important when assigning SMs to requesting UEs. This factor is only considered here when user *k* currently does not have a video segment scheduled over the cellular channel. Hence, the factor σ_k is used. If user *k* is currently not being sent a video segment over the cellular channel, σ_k will be set to 1. Otherwise, σ_k will be set to 0 so that $p_{k,d}[n]$ is not considered in the scheduling metric. This is to consider only the reported value of the buffer length and the duration of video sent over the cellular channel, in this case, and avoid abandoning this user when allocating cellular resources if it has a low playout buffer. $p_{k,d}[n]$ significantly reduces the scheduling metric of the user *k*. If the video segment that is currently scheduled over cellular resources precedes the one transmitted over the D2D channel, $p_{k,d}[n]$ might falsely indicate that user *k* has a long playout buffer, which results in long delay.

From the above, we can see that the metric used for BB scheduling considers the reported playout buffer length as well as an estimation of the length of recently transmitted video contents from all sources. As such, the scheduler will be always aware of the current playout buffer length at the UEs, and hence, avoid rebufferings by allocating more resources to users with imminent playout buffer stalling. In a cell with high number of UEs competing for the shared and limited cellular resources, the first goal should be decreasing the number of rebufferings, as it has the highest impact on video streaming QoE.

In the context of DABAST, the BB scheduler is aware of the playout buffer length of users, and hence, it will save the cellular resources for users who get their video segments exclusively through the BS over cellular resources. The scheduler can employ the information on the playout buffer length to further utilize the D2D channel. By refraining from sending video segments over cellular resources to users with playout buffer length higher than a certain threshold, it will force more video segments to be transmitted over the D2D channel.

8.2 SM assignment

In DABAST, it is realistic to assume that SMs accept up to a certain number of concurrent assistance requests. As such, the BS needs to select which SMs are used to send which video segments. Figure 8.2 shows an example where there are 2 SMs and 3 requesting UEs in a cluster. SM_1 can assists with sending video segment B to UE_2. However, the BS might need to decide whether to assign SM_1 to send segment A to UE_1 or UE_3. This is just a simple example that shows that the BS should decide which SM should send video segments to which requesting UE.

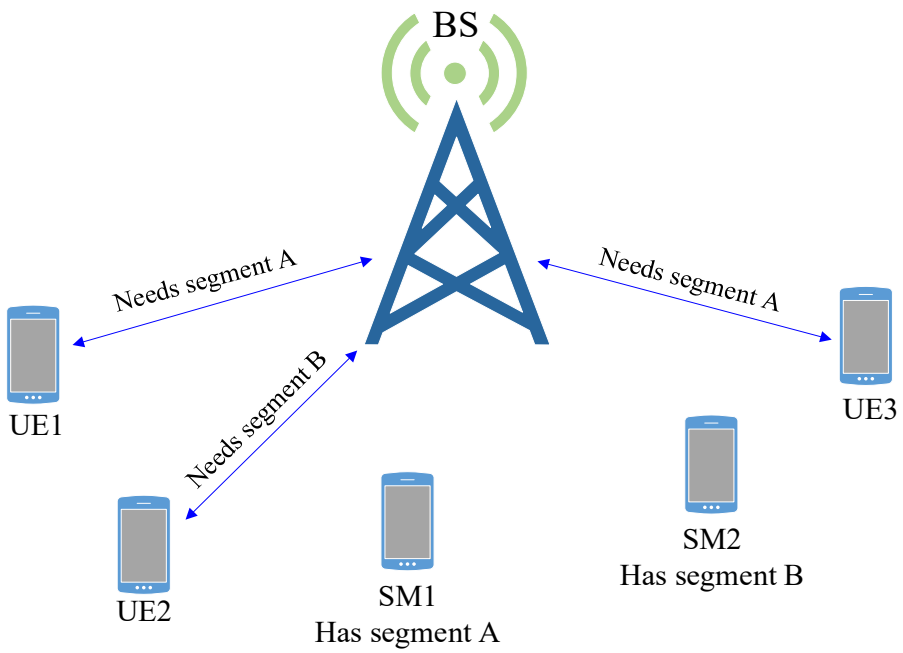

Figure 8.2. Example of the SM assignment problem.

We refer to this as the SM assignment problem. Here, we discuss how SMs are assigned the tasks of video segment transmission to requesting UEs in case of RR, PF, and BB cellular resource scheduling.

In the case of RR scheduling, every time SM assignment is performed, the BS goes through the active UEs in a RR fashion and looks for a UE that needs a video segment that is available in the distributed cache with an SM that is willing to assist. This means that SM assignment is performed in RR as well. In the case the BS finds more than one SM that is available to send a video segment to a requesting UE, the BS selects in a way that achieves load balancing among the SMs, i.e., the BS picks the SM that provided the least assistance so far. This way, the BS tries to achieve fairness for both the requesting UEs as well as the assisting SMs.

In the case of PF scheduling, the recent average throughput of the requesting UEs and the

average data rate between SMs and requesting UEs can be both used for allocating SMs to requesting UEs. This means that PF is used for SM assignment in a similar way cellular resources are allocated. Every time SM assignment is performed, the BS goes through the requesting UEs and selects the one with the highest ratio of the average data rate to an SM over the recent average throughput of that UE. This way, we assume that UEs periodically report to the BS an estimation of the average data rate that can be achieved if data is transmitted from a certain SM to this requesting UE as proposed in [7]. In the case the number of requesting UEs is high, this would cause much overhead. As such, the recent average throughput only can be utilized for SM assignment. This means that the BS goes through the requesting UEs in an ascending order based on the recent average throughput, and every time the BS finds a requesting UE that needs a cached segment with available SM, it will send assistance request to that SM. As with RR, in the case the average data rate between the UEs and SMs are not utilized, load balancing-based approach can be used to select among multiple available SMs.

In the case of BB scheduling, the UEs signal their playout buffer length to the BS. Since this valuable information is available at the BS, it can be further utilized for SM assignment. In this case, the BS assign available SMs, giving more priority to UEs who have a low playout buffer. This further decreases the possibility of playout buffer depletion and consequently reduces the number of rebufferings for users in the cell, which potentially increases their QoE. When performing SM assignment, all the factors in (8.4) will be considered, i.e., $p_{k,d}[n]$ is always utilized, and hence is calculated as follows,

$$p_{k,d}[n] = N_d \times w \times L. \tag{8.8}$$

In the next chapter, we propose a more sophisticated SM assignment approach that has more than one objective to further increase the QoE of users in the cell.

8.3 Performance evaluation

We ran various simulations to evaluate the performance of DABAST in terms of video streaming QoE with all the scheduling algorithms discussed above (RR, PF, and BB). We use DABAST-CSVD here as our case study, and hence, we drop the name of the caching and distribution algorithm, i.e., we use the name DABAST. We measure the same QoE metrics used in the previous chapter. The simulation setup in the previous chapter (Table 7.1 and Table 7.2) was also used here. Table 8.1 shows the mean values of the measured QoE metrics along with the MoE values for 95% confidence interval. The results in table 8.1 are for 500 UEs in the cell, Zipf exponent of 1.5, and 500 videos. The average of all the values from a simulation run is calculated. The values in Table 8.1 show the mean of all the average values from 50 simulation runs.

When comparing the results for DABAST with RR and PF scheduling, it can be seen that PF scheduling reduces the average number of rebufferings by 0.2848 (16.49% reduction), which is a considerable improvement. Employing PF scheduling also caused a slight improvement in the continuity index and in the average video bit rate. With PF scheduling, the scheduling metric considers the ratio of the instantaneous data rate of the UE to the recent average throughput of the UE. The recent average throughput considers data transmitted over both cellular and D2D resources. By maintaining the recent average throughput of each UE, we consider the video data that is transmitted over both the

cellular and the D2D channel. Hence, for UEs that received segments over the D2D channel, the recent average throughput will be relatively high, which consequently leads to favoring UEs with lower recent average throughput.

Table 8.1. Simulation results (RR vs. PF vs. BB).

	DABAST-RR		DABAST-PF		DABAST-BB	
	Mean	MoE	Mean	MoE	Mean	MoE
Rebufferings	1.7272	0.0164	1.4424	0.0127	0.8787	0.1195
Cont. index	0.8699	0.0001	0.8923	0.0009	0.9258	0.0079
Initial delay(sec)	28.654	0.4821	31.588	0.3159	28.376	0.7506
Video bit rate (kbps)	430.16	1.3694	433.47	0.8354	428.72	1.6469

Usually, these are UEs with low playout buffer and UEs that receive their video segments exclusively over the cellular channel. In this scenario where many UEs are sharing the cellular channel, it would be beneficial to dedicate the limited cellular resources to UEs that can only get their video segments over the cellular channel. This will increase the cellular resources for these users, and further relax the bottleneck of the RAN, which reduces the number of rebufferings and increases the continuity index.

Although PF scheduling keeps track of the recent average throughput for each UE, which helps in reducing the average number of rebufferings, it is oblivious to the current playout buffer length of users. There are many cases where a high value of the recent average throughput might falsely indicate a high playout buffer length value. A general case is when a video segment with high video bit rate is transmitted to a UE. Although in such case the UE experienced high recent throughput, this was used to send relatively less video content (in seconds) due to the high video bit rate of the segment. There is a

different and more complicated case that is specific to DABAST. With DABAST, a video segment might be sent over the D2D channel when found in the distributed cache, although the previous video segment is still being transmitted over the cellular channel. This speeds up transmission of video segments and keeps the playout buffer length consistently high. It is worth explaining that in such case, the previous segment was set for transmission over the cellular channel due to unavailability of SMs at the time this decision was made. In this case, the UE will have a high recent average throughput because a segment is transmitted over the D2D channel. Hence, the PF scheduling metric for this UE will be low, although the previous video segment is still being transmitted over the cellular channel. A low scheduling metric means that the PF scheduler will give low priority to this UE when allocating cellular resources. This usually does not result in rebuffering because such UEs (playing videos cached in the SMs) have relatively higher playout buffer as most segments are delivered over the D2D channel. Hence, the segment transmitted over the cellular channel arrives before consumption of the playout buffer. However, this might result in high initial delay when the UE is awaiting the initial video segments needed to start playout. This explains why the average initial delay is higher for DABAST with PF scheduling.

Table 8.1 shows that employing BB scheduling achieved significant improvement in terms of rebuffering. BB scheduling achieved 49.13% and 39.08% reduction in the average number of rebuffering over RR and PF, respectively. Table 8.1 shows that BB scheduling also significantly increased the continuity index. BB scheduling achieved these improvements in terms of rebuffering, while achieving a slight improvement in the average initial delay. With BB scheduling, the scheduler considers the reported playout

buffer length as well as an estimation of the length of recently transmitted video contents. As such, the scheduler will be always aware of the current playout buffer length at the UEs, and hence, avoid rebufferings by allocating more resources to users with low playout buffer. This explains the improvement achieved by BB scheduling in terms of the number of rebufferings and the continuity index. Because BB scheduling keeps track of the actual playout buffer length, it will allocate resources to a UE if it currently has a relatively low playout buffer, even if the next segment is transmitted over the D2D channel. As such, BB scheduling does not increase the initial delay as with PF scheduling.

Table 8.1 shows that BB scheduling results in a very slight reduction in the average video bit rate. BB scheduling has 428.72 kbps average video bit rate, which is only 0.33% and 1.1% reduction when compared to RR and PF scheduling, respectively. This is a negligible reduction that is not even noticeable by the end user. This reduction is expected as BB scheduling utilizes more of the cellular resources to help users who get their video segments exclusively through the BS, and hence fewer segments will be sent with higher video bit rate to users with higher playout buffer, i.e., users who get some of their segments over the D2D channel.

Figures 8.3 and 8.4 depict the histogram and the relative frequency histogram, respectively, of the number of rebufferings for DABAST with RR, PF, and BB resource allocation. When comparing RR and PF scheduling, one can notice that with PF scheduling, more video streams have 0, 1, and 2 rebufferings, and fewer video streams have 3 and 4 rebufferings. This is because when allocating cellular resources, the PF scheduler favors UEs that have low recent average throughput. Usually, these are UEs with low playout buffer length and UEs that receive their video segments exclusively

over the cellular channel. This increases the cellular resources allocated for these users and increases avoidance of rebufferings.

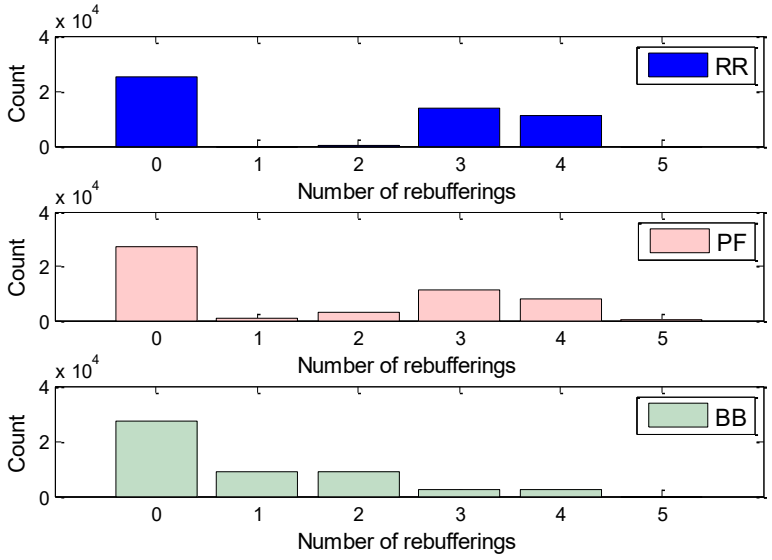

Figure 8.3. Histogram of the number of rebufferings for DABAST with RR, PF, and BB resource allocation.

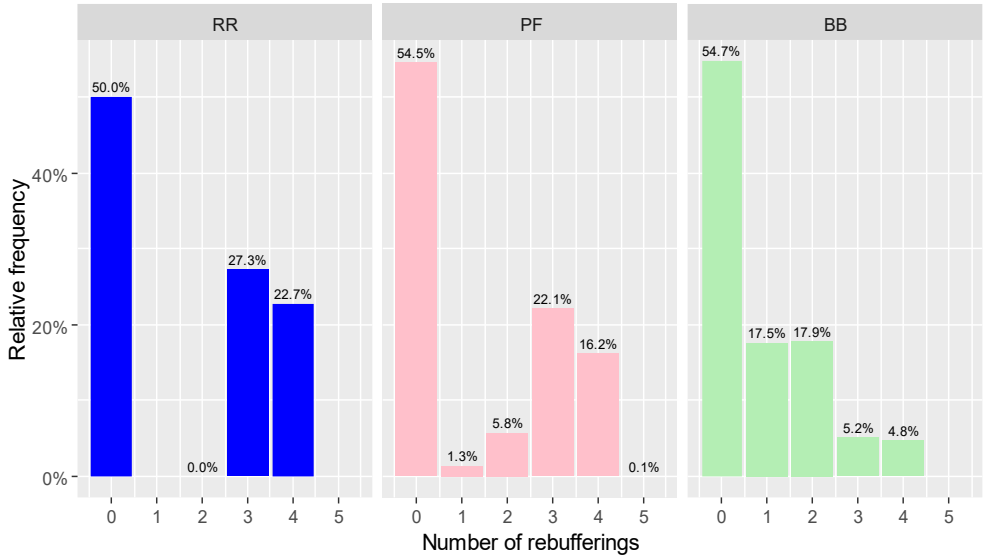

Figure 8.4. Relative frequency histogram of the number of rebufferings for DABAST with RR, PF, and BB resource allocation.

One can see from Figure 8.3 and Figure 8.4 that with PF scheduling, there is a very small number of streams that have 5 rebufferings. As discussed above, this is because the PF

scheduler is oblivious to the current playout buffer length of the users in the cell, and in some cases the higher value of the recent average throughput does not necessarily indicate a longer playout buffer. As such, the PF scheduler might ignore such users and cause them to experience 5 rebufferings.

Figures 8.3 and 8.4 show that BB scheduling significantly reduces the number of streams with 3 and 4 rebufferings, and eliminates the case of 5 rebufferings. This explains the significant improvement achieved by BB scheduling in terms of the average number of rebufferings. One can see from the figures the importance of the playout buffer length awareness at the scheduler. This allows the scheduler to avoid video buffer depletion, as much as possible, and hence reduce the number of rebufferings.

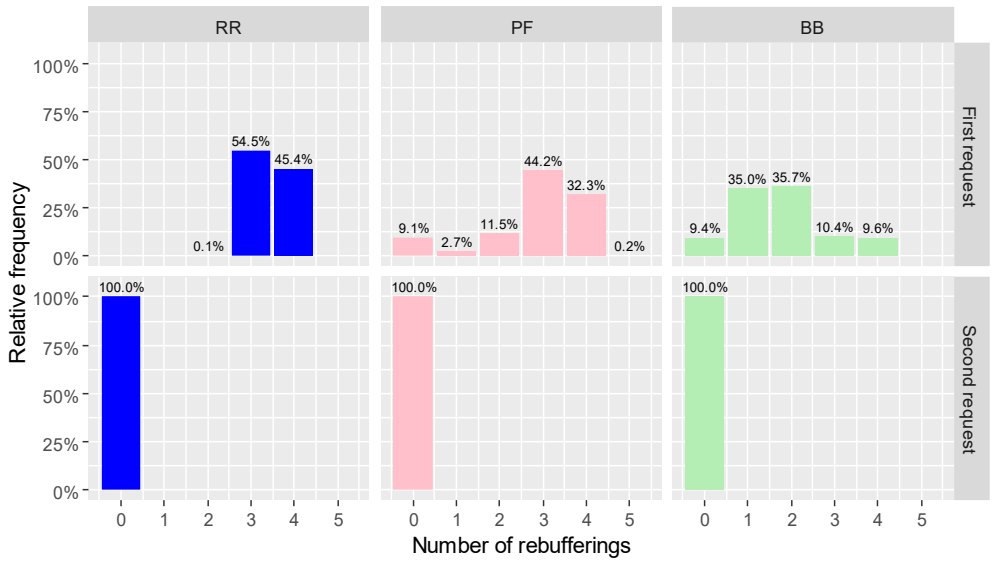

Figure 8.5. Relative frequency histogram of the number of rebufferings in each request for DABAST with RR, PF, and BB resource allocation.

Figure 8.5 shows the relative histogram for the number of rebufferings of each request for DABAST with RR, PF, and BB resource allocation. The figure shows that with RR scheduling, almost all the video streams in the first request have 3 or 4 rebufferings. With

BB scheduling, on the other hand, 80.1% of the video streams have 0, 1, or 2 rebufferings, and only 19.9% have 3 or 4 rebufferings.

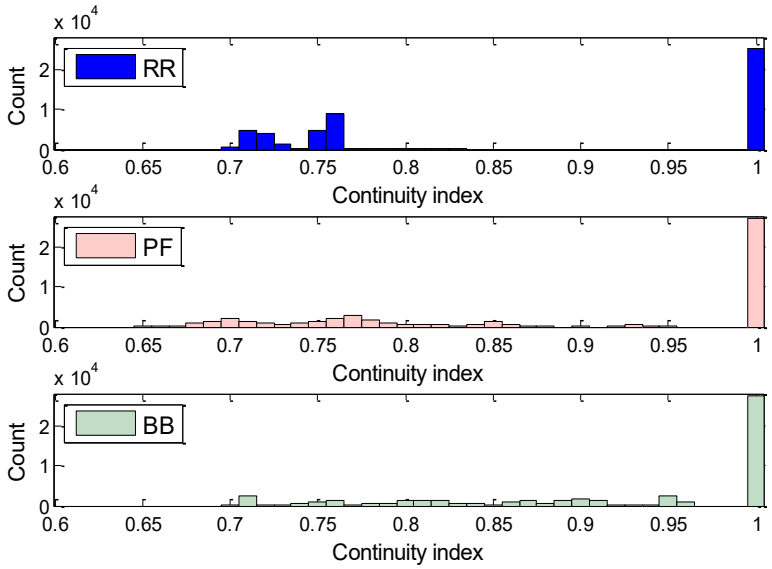

Figure 8.6. Histogram of the continuity index for DABAST with RR, PF, and BB resource allocation.

Figure 8.6 shows the histogram of the continuity-index for DABAST with RR, PF, and BB resource allocation. As can be seen from the figure, although PF scheduling increased the continuity index value for many video streams when compared to RR scheduling, there is a small number of streams that have continuity index values less than 0.7, which is the lowest value with RR scheduling. These are the users who experience 5 rebufferings, as explained above. With BB scheduling on the other hand, the continuity index value for many streams have increased, and the minimum values did not decrease (still at 0.7).

8.4 Summary

In this chapter, DABAST with QoE-aware resource allocation is presented. The proposed method for resource allocation in DABAST takes into account the reported playout buffer

length in the UEs as well as estimation of the video duration recently transmitted to UEs over both the cellular and D2D channels (from all sources). The proposed scheduling algorithm takes into account all the information above to allocate the cellular resources in a way that improves the QoE for the users in the cell. Results show that the QoE-aware scheduling algorithm, BB, improves the achieved video streaming QoE for users in the cell when compared to QoE-oblivious algorithms (e.g., PF scheduling).

Chapter 9

QoE-aware DABAST: High rate caching and SM assignment optimization

In the previous chapter, we introduced QoE awareness to cellular resource allocation in DABAST. Results have shown that the implemented BB scheduling algorithm, which allocates resources based on the playout buffer length at the UEs, achieves a significant reduction in the number of rebufferings in the cell. This is very important in cases where the number of UEs in the cell is high enough so that many UEs are experiencing rebufferings. In this chapter, we consider cases where the available resources are enough to avoid rebufferings. We aim to further increase the utilization of the D2D channel in such cases to improve video streaming QoE for users in the cell by improving their video bit rate. In the first section, we present the first proposed technique; high rate caching. Afterwards, we present the second proposed technique, which is SM assignment optimization. Finally, we evaluate through simulations the performance achieved with these techniques in terms of video streaming QoE.

9.1 High rate caching

As reported by many studies, video rebuffering has the highest impact on video streaming QoE, and hence, it should be avoided as much as possible. As such, when the traffic load is very high, reducing the number video rebufferings should be a priority. In these situations, video segments will be downloaded with low video bit rate to avoid rebufferings. However, when the traffic load is lower, and users are not experiencing

rebufferings, it would be very beneficial to send popular videos with high video bit rate. If segments of popular videos are cached with high video bit rate, they will be sent later to requesting users over the D2D channel, which will considerably increase video bit rate for users who receive these segments later from the distributed cache, and hence, increase their video streaming QoE.

We propose DABAST with high rate caching, to further improve video bit rate, and improve video streaming QoE. High rate caching is implemented by updating the operation of the CSVD algorithm, to consider one more case. We refer to this case as the *high rate caching* case. This case is implemented as follows:

1) The first condition that is needed to consider high rate caching is "low" traffic load. Otherwise, high rate caching will further increase the number of rebufferings in the cell and reduce video streaming QoE. This can be decided based on the number of rebufferings. For instance, the high rate caching mode can be activated in the case that none of the users in the cell are experiencing video rebuffering, or if the number of rebufferings is below a certain threshold.

2) The second condition to employ high rate caching is if video segments are sent to an SM. In this case, segments of the video will be cached, and hence, will be sent later to requesting UEs in a high video bit rate. On the other hand, if the video is not going to be cached, only the video bit rate of this stream will increase, and this taken risk of sending a video with high video bit rate will not result in a significant reward. Because of this, video segments are only sent in a video bit rate that is higher than the requested rate, if the segments are sent to an SM. This

way, the high video rate segments are cached, and utilized by consequent requests.

3) High rate caching is only employed if the requested video is popular. This is explained as in the previous condition. If the video that is transmitted and cached with high video bit rate is not popular, the taken risk of sending a video with high video bit rate will not result in a significant reward. This is because if the file is not popular, it may not be requested often after this time, and consequently, will not be utilized by later requests. A video file can be considered popular enough for high rate caching if it has been requested more than a certain number of times recently.

4) To avoid increasing the initial delay, the first few segments are always transmitted with a low video bit rate, even in high rate caching. The goal of high rate caching is to increase the video bit rate without causing considerable degradation to other video streaming QoE metrics.

High rate caching ensures that video segments of popular videos are cached with high video bit rate. This will diversify the content of the distributed cache in terms of video bit rate, i.e., some segments might be cached in more than one video bit rate. As such, the BS needs ensure that high rate segments are fully exploited, and at the same time, make sure that no degradation is induced to other video streaming QoE metrics. This can be achieved by carefully assigning SMs the tasks of sending video segments. This means that high rate caching introduces more complexity to the problem of SM assignment.

One way to implement SM assignment in this case is by employing the same approach used with BB scheduling in the previous chapter. That approach assigns available SMs,

giving more priority to UEs who have a low playout buffer, to minimize the number of rebufferings, and in the case more than one SM is available, the decision will be made based on load balancing among SMs. Never the less, this approach does not take into account video bit rate of the cached segments. Such a video bit rate oblivious approach will not fully utilize the high video bit rate segments that are available in the distributed cache of the network.

In the next section, we propose a better approach that takes into account both the playout buffer length at the clients and the video bit rate of the cached segments.

9.2 SM assignment optimization

Here, we consider the problem of SM assignment in the case where there are many UEs requesting video segments that are cached with multiple video bit rates. An SM caches a segment in one video bit rate, but the segment might be available in a different video bit rate at another SM. An optimal SM assignment approach, in this case, would maximize the achievable video bit rate in the cell, without causing a considerable increase in the number of rebufferings or in the initial delay. This should be achieved under the condition that an SM only accepts up to a maximum number of concurrent requests. We also assume that the BS sends up to a maximum number of assistance requests each time SM assignment is performed.

Figure 9.1 shows an example of the case described above. The figure depicts 3 requesting UEs and 3 SMs. To maximize the achieved video bit rate. The SM assignment should result in SM_2 sending segment A to UE_1, and SM_3 sending segment B to UE_2. However, considering that UE_3 has a low playout buffer, the SM assignment should result in SM_1 sending segment B to UE_2, SM_2 sending segment A to UE_1, and SM_3 sending segment C to UE_3 to avoid imminent rebuffering that could be experienced by UE_3.

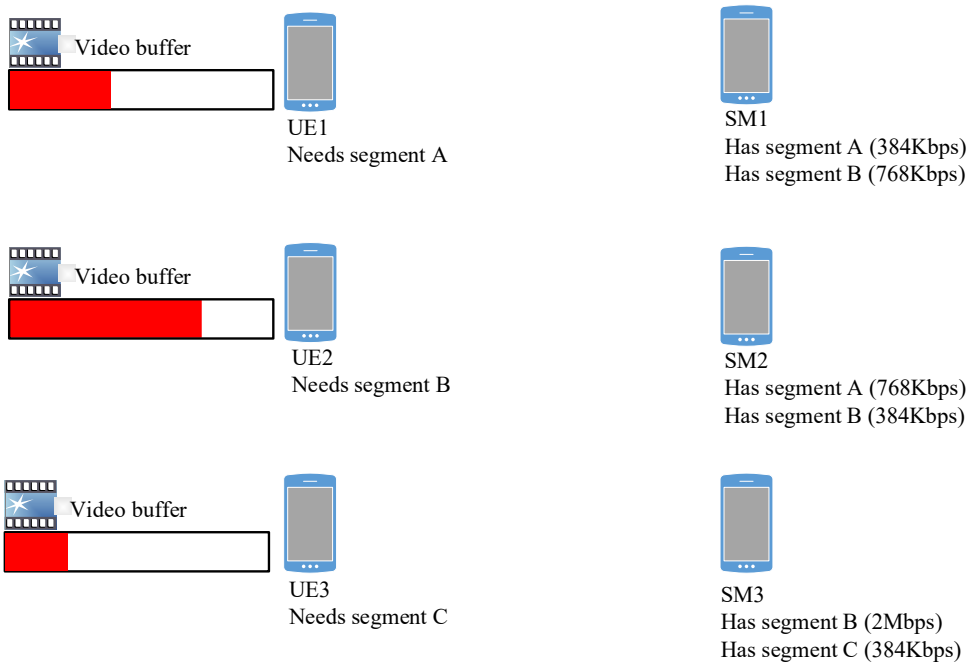

Figure 9.1. SM assignment example.

9.2.1 SM assignment optimization problem

SM assignment is performed by the SM assignment module at the BS, which runs an optimization algorithm to find the best assignment. We formulated the above SM assignment problem as a Mixed Integer Linear Programming (MILP) problem as follows,

$$\max \sum_{c \in C} \left(\sum_{i \in I_c} \sum_{j \in J_c} x_{i,j} \cdot R_{i,j} - \sum_{i \in I_c} \left(1 - \sum_{j \in J_c} x_{i,j} \right) \cdot e^{\alpha / L_i} \right), \tag{9.1}$$

s.t.

$$\sum_{j \in J_c} x_{i,j} \leq 1, \ \forall \ i \in I_c, \ \forall c \in C, \tag{9.2}$$

$$\sum_{i \in I_c} x_{i,j} \leq D_j, \ \forall \ j \in J_c, \ \forall \ c \in C, \tag{9.3}$$

$$\sum_{c \in C}\left(\sum_{i \in I_c}\sum_{j \in J_c} x_{i,j}\right) \leq A, \tag{9.4}$$

$$x_{i,j} \in \{0,1\},\ \forall (i,j) \in I_c J_c,\ \forall\ c \in C, \tag{9.5}$$

where,

- $x_{i,j}$ is the binary optimization variable for the *ith* UE and *jth* SM in the cluster, and indicates whether or not SM_j is assigned to UE_i, i.e., $x_{i,j}$ is 1 if SM_j is assigned to requesting UE_i (SM_j is sending a video segment to UE_i) and 0 otherwise
- C is the set of clusters in the cell
- I_c is a set of selected requesting UEs in cluster c. Every UE in I_c should have at least one SM in that cluster that is available to provide the next video segment
- J_c is a set of selected SMs in cluster c. Every SM in J_c should be available for assistance and has at least one video segment that is currently requested by one of the requesting UEs in I_c
- A is the maximum number of assistance requests that can be sent each time SM assignment is performed
- $R_{i,j}$ equals the video bit rate for the video segment cached at SM_j and requested by UE_i if the segment is available at SM_j, and it equals -M (very large negative number) otherwise
- D_j is the maximum assistance requests that can be concurrently assigned to SM_j
- α is an optimization parameter
- L_i is the current playout buffer length of requesting UE_i

Equation (9.1) shows the objective function to be maximized. The objective function is

composed to two parts. The first part (before the minus sign) is to maximize the aggregate video bit rate. This can be achieved by assigning to each requesting UE the SM that is caching the requested segment with the highest video bit rate, which results in setting the binary variable of that pair, $x_{i,j}$, to 1. The second part of the objective function (after the minus sign) aims to minimizing the number of rebufferings, by ensuring that the playout buffer of the UEs is above a certain level. The parameter α decides how aggressive the optimizer is in favoring playout buffer depletion avoidance over maximizing the aggregate video bit rate. It controls the playout buffer level the optimizer tries to maintain before it starts assigning SMs to UEs solely to maximize the average video bit rate.

To understand the second term of the objective function, let us start by noting that each UE is assigned a maximum of one SM each time SM assignment is performed (first constraint in (9.2)). In the case there is a UE with a low playout buffer length, the small value of L_i will significantly increase the value of the exponential term. If an SM is assigned to this UE, subtracting the total number of SMs assigned to this UE (which is 1) from 1 will result in 0 and will cancel the high value of the exponential term, which maximizes the overall objective function. However, if no SM is assigned to this UE, the value of the exponential term will significantly increase the value of the second part (after the minus sign) and reduces the value of the objective function. When all the UEs have a relatively high playout buffer length, the values of the exponential term will be small, and hence, the optimizer will focus only on the first term of the objective function which maximizes the aggregate video bit rate.

From the above, we can see that this results in dynamically maximizing video bit rate.

When there are UEs with low playout buffer that is below a certain threshold (decided by the value of α), increasing the playout buffer length of these UEs will have higher priority than maximizing video bit rate. However, if the playout buffer length of the UEs is above that threshold, the highest priority will be to maximize video bit rate. In addition to maximizing the aggregate video bit rate dynamically while avoiding rebuffering, A considerable increase in the initial delay can also be avoided by controlling the value of α. This is because further increase to the value of α gives more priority to filling up the playout buffer of clients.

The first constraint which is imposed by equation (9.2) states that every time SM assignment is performed, a maximum of 1 SM can be assigned to any UE. The second constraint shown in equation (9.3) ensures that the number of assists sent to any SM does not exceed the maximum concurrent assists of that SM. The third constraint, which is imposed by equation (9.4) states that every time SM assignment is performed, the number of assists should not exceed a certain value. Equation (9.5) specifies the variable bounds of the optimization problem.

9.2.2 An alternative formulation

In the following, we present an alternative formulation of the SM assignment problem. this formulation represents the same problem and provide the same solutions. We presented the first formulation above as it is easier to understand. However, the following formalization has fewer decision variables, as it only has decision variables regarding the UE-SM combinations where the SM has the segment requested by that UE, instead of having a decision variable for every requesting UE and helping SM who are in the same cluster. Furthermore, this formalization is easier to implement with our simulator, as

shown in the next section.

Let us consider a cell with multiple clusters. C represents the set of clusters in the cell. Each cluster has a set of helping SMs, M_c. M_c contains SMs who have cached video segments that are currently requested by some UEs in the same cluster. A cluster also has a set of requesting UEs, Q_c, which contains the requesting UEs, for which, the currently requested segment can be provided by at least one SM in M_c. If G is the set of all video segments that can be requested, i.e., all the video segments of the current video streams, then each SM, $m \in M_c$, is caching a subset of the video segments, $G_m \subseteq G$. Moreover, each requesting UE, $q \in Q_c$, is currently requesting a segment $g_q \in G$. For each cluster, $c \in C$, S_c is the set of ordered pairs that represents the UE-SM combinations, in which, the SM has the segment requested by that UE. S_c can be represented as follows,

$$S_c = \left\{ \langle q, m \rangle \mid q \in Q_c \wedge m \in M_c \wedge g_q \in G_m \right\}. \tag{9.6}$$

$S'_{c,z}$ is a subset of S_c, that contains only the ordered pairs that has z as the first item in the pair, where $z \in Q_c$. $S'_{c,z}$ can be expressed as,

$$S'_{c,z} = \left\{ \langle q, m \rangle \mid q = z \wedge m \in M_c \wedge g_q \in G_m \right\}. \tag{9.7}$$

Similarly, $S''_{c,t}$ is a subset of S_c, that contains only the ordered pairs that has t as the second item in the pair, where $t \in M_c$. $S''_{c,t}$ can be expressed as,

$$S''_{c,t} = \left\{ \langle q, m \rangle \mid q \in Q_c \wedge m = t \wedge g_q \in G_m \right\}. \tag{9.8}$$

The SM assignment problem can be formulated as an MILP problem as follows,

$$\max \sum_{c \in C} \left(\sum_{s \in S_c} x_s \cdot R_s - \sum_{z \in Q_c} \left(\left(1 - \sum_{s \in S'_{c,z}} x_s \right) \cdot e^{\alpha / L_z} \right) \right), \tag{9.9}$$

s.t.

$$\sum_{s \in S'_{c,z}} x_s \leq 1, \ \forall \ z \in Q_c, \ \forall \ c \in C, \tag{9.10}$$

$$\sum_{s \in S''_{c,t}} x_s \leq D_t, \ \forall \ t \in M_c, \ \forall \ c \in C, \tag{9.11}$$

$$\sum_{c \in C} \left(\sum_{s \in S_c} x_s \right) \leq A, \tag{9.12}$$

$$x_s \in \{0,1\}, \forall s \in S_c, \forall c \in C, \tag{9.13}$$

where,

- x_s is the binary optimization variable for the *sth* UE-SM combination. It indicates whether this UE-SM combination will be selected, i.e., x_s is 1 if the SM in this combination is assigned to the requesting UE in this combination, and 0 otherwise.
- A is the maximum number of assistance requests that can be sent each time SM assignment is performed.
- R_s is the video bit rate of the segment for combination s, i.e., the video segment cached at the SM and requested by the UE in the combination.

- D_t is the maximum assistance requests that can be concurrently assigned to SM t.
- α is an optimization parameter.
- L_z is the current playout buffer length of requesting UE z.

The above SM assignment optimization problem is an MILP with $\sum_{c \in C} |S_c|$ variables, where S_c is the set of UE-SM combinations in cluster c. S_c only contains the combinations in which, the SM can provide the current video segment requested by that UE. If S_c contains many combinations, then solving the above problem might not be feasible in a TTI time. However, SM assignment does not have to be performed every TTI like cellular resource allocation. As SM assignment in DABAST is done on a segment-by-segment basis, the SM assignment optimization problem can be performed on a scale of tens of milliseconds. Fortunately, there are many commercial solvers that can solve the above problem quickly. As explained in the next section, we employ Gurobi, an optimization solver [99]. Gurobi uses the Linear Programming (LP)-based branch-and-bound algorithm to solve MILP problems [100]. Furthermore, it employs many techniques to speed up execution time of the LP-based branch-and-bound algorithm, such as pre-solving, cutting planes, heuristics, and parallelism. We measured the execution time of the SM assignment optimization problem with Gurobi for the simulations in Section 9.4. The used machine has a Quad-core Intel i7 processor with a speed of 3.6 GHz x 8 threads. Measurements have shown that the maximum execution time of the problem was 10 ms, which is adequate for the time scale of SM assignment (tens of milliseconds as mentioned above).

9.3 Integrated modeling, simulation, and optimization

As mentioned in the previous chapters, we used the CD++ toolkit to implement our LTE-A network DEVS models, which is an open-source simulation software written in C++ and implements the DEVS abstract simulation technique. The simulation engine tool of CD++ is built as a class hierarchy. With CD++, atomic models are developed using the C++ programming language and can be incorporated into the class hierarchy. However, CD++ does not include a tool for optimization problem solving. To implement the SM assignment module in our model, we developed our CD++ toolkit to have optimization capabilities. To achieve this, we integrated Gurobi into CD++.

9.3.1 The Gurobi optimizer

Gurobi is an optimization solver [99], [101] that can be used for LP, Quadratic Programming (QP), Quadratically Constrained Programming (QCP), MILP, Mixed-Integer Quadratic Programming (MIQP), and Mixed-Integer Quadratically Constrained Programming (MIQCP). Gurobi provides support for many programming languages such as C, C++, Java, and Python. It also provides matrix-oriented interfaces that can be useful for languages such as C and MATLAB [99], [101]. Furthermore, it has links to many modeling languages. In this section, we show how we integrated Gurobi into our DEVS simulator and used it to implement the SM assignment module in our DEVS model.

Figure 9.2 shows a simplified UML diagram for the C++ implementation of Gurobi [102]. The diagram shows the main classes needed to create and run an optimization model. The components needed to create a model and shown in Figure 9.2 are briefly presented here. For further information on the classes and functions available in the C++

Gurobi optimizer, the reader is referred to the Gurobi reference manual [102]. The class *GRBEnv* is used to create a Gurobi optimization environment, which is a container for all the optimization data in the program. An object of the *GRBEnv* must be created at the first step and can be used with all the optimization models in the program [102]. Different settings are available for the optimization environment. A local environment can be created where the Gurobi models are solved on the local computer. Alternatively, Gurobi models can be solved on a on a Gurobi server or on a cloud server by creating a client environment for a Gurobi compute server or an instant cloud environment, respectively.

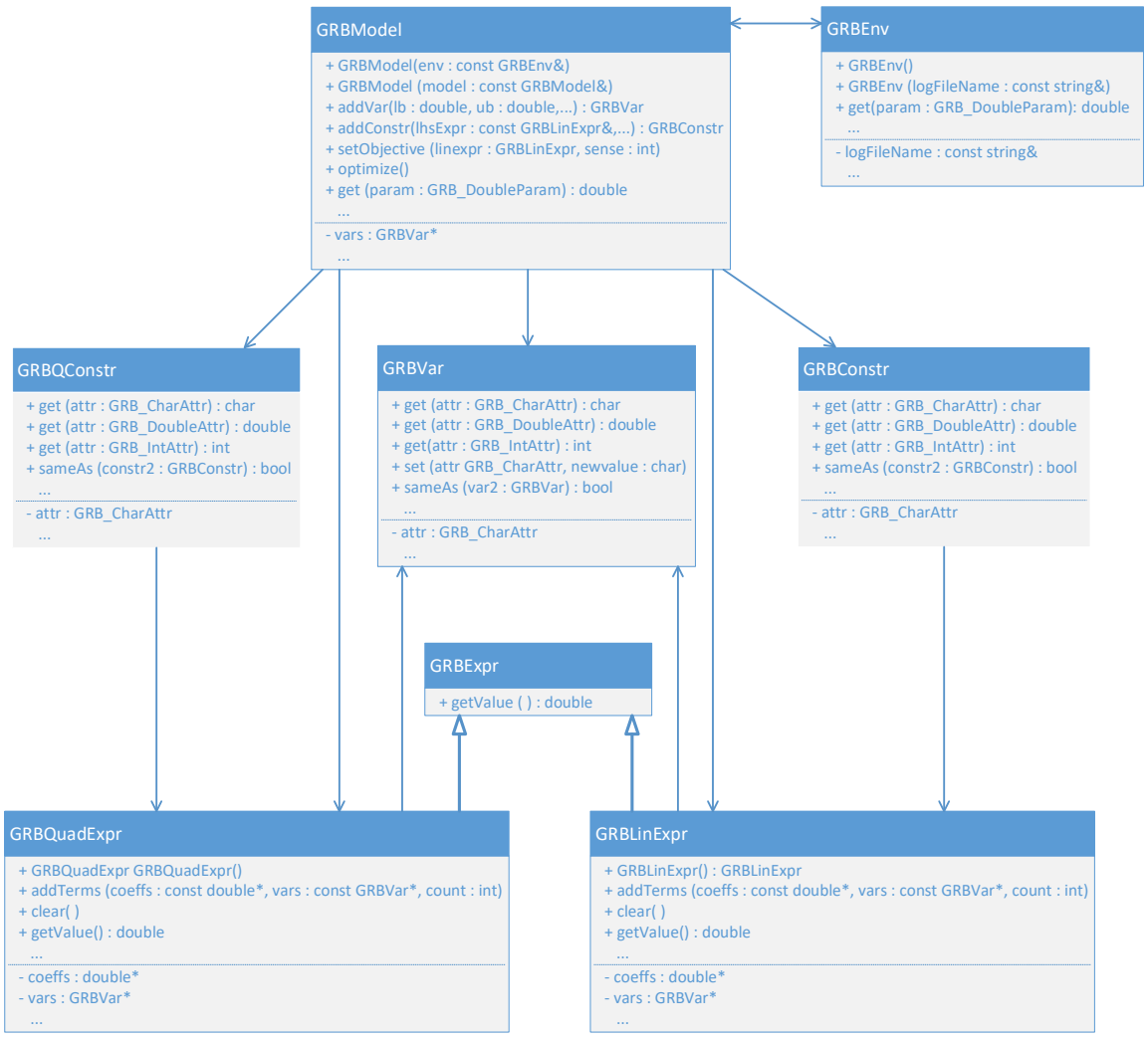

Figure 9.2. Simplified UML diagram of the C++ based Gurobi optimizer.

The class *GRBModel* is used to create a Gurobi optimization model [102]. As can be seen from the constructor, a *GRBModel* should specify the Gurobi environment for each model. This constructor creates an empty model where the variables, objective, and constraints can be added later. The diagram also shows that a copy constructor is available to create a new GRB model form an existing one. The function *addVar* is used to add a single decision variable to the model. It takes many arguments such as the upper and lower bounds for the added decision variable. Other functions to add multiple decision variables at once are also available. The function *addConstr* is used to add a linear constraint to the Gurobi model. It takes arguments such as the left-hand side and right-hand side linear expressions, the sense of the constraint (equals, greater than, etc.), and the name of the constraint. There are various functions available to add constraints to the model, and many signatures are provided for many of these functions. For instance, the function *addConstrs* can be used to add more than one constraint at once, while the function *addQConstr* can be used to add a single quadratic constraint to a model. The function *setObjective* is obviously used to set the objective of the optimization model. It takes as arguments a linear expression and an integer that represents the sense of the objective function (GRB_MINIMIZE to minimize and GRB_MAXIMIZE to maximize). As most other functions, different signatures are available to this function. Once the variables, objective function, constraints, and variable bounds are all set, the function *optimize* can be called on an object of the *GRBModel* class to optimize the model. After optimizing the model, the function *get* can be used to obtain the values of the decision variables.

The class *GRBVar* is used to represent decision variables used in the optimization model. This class does not have a constructor and objects of this class are created with the function *addVar* in the *GRBModel* class. This is to make sure that each object of this type is associated with a *GRBModel* object. Objects of the class can have various attributes. As can be seen in the UML diagram, getters are available to obtain the values of these attributes such as the values of the decision variable.

GRBExpr is an abstract class that is extended by the *GRBLinExpr* and *GRBQuadExpr* subclasses. The *GRBLinExpr* objects represent linear expressions. Each object of this class contains a list of pairs. Each pair consist of a *GRBVar* object and a coefficient. Overloaded operators are available to build Gurobi linear expressions. For instance, if x and y are objects of the *GRBVar* class, then $x+y$ is a linear expression. Furthermore, overloaded operators can be used to build linear expressions from *GRBLinExpr* objects. For example, if *linExpr1* and *linExpr2* are both objects of the *GRBLinExpr* class, then $linExpr1 + linExpr2$ is also a linear expression. Adding expressions to variables is also possible (e.g., $linExpr1 + x$). Linear expressions are used to form linear objective functions and linear constraints to impose on optimization models. The *GRBQuadExpr* class is used to create quadratic expressions, which are used to form quadratic constrains and quadratic objective functions.

Objects of the *GRBConstr* class are used to impose linear constraints on the optimization model. Every object of the *GRBConstr* must be associated with a *GRBModel* object. To ensure this condition, the *GRBConstr* class does not have a constructor, and objects of

this class are created by calling the method *addConstr* on a *GRBModel* object. Getters are available to obtain the attributes for the *GRBConstr* objects.

Objects of the *GRBQConstr* class represent Gurobi quadratic constraints that can be added to Gurobi optimization models. As with the *GRBConstr*, objects of this class are created by calling the method *addQConstr* on a *GRBModel* object.

9.3.2 Integrating CD++ and Gurobi

In this section, we present how CD++ and Gurobi were integrated to provide an environment for modeling, simulation, and optimization. This is achieved by building the needed Gurobi optimization models, running them during simulations, and using the results from the optimized Gurobi models to take optimization decisions and proceed with simulations.

To implement this integration, an Integrator module was built which consists of two components, *CD++2Gurobi* and *Gurobi2CD++*. The flowchart in Figure 9.3 shows in detail how the components above are used to implement the integration process. As one can see in the flowchart, there are four main components in the integrated environment, namely, the CD++ simulator, both components of the integrator (*CD++2Gurobi* and *Gurobi2CD++*), and Gurobi. Whenever the SM assignment optimization is needed in the simulation, the *CD++2Gurobi* routine in the integrator will go through the active streaming sessions. For each session, UE-SM combination(s) might be built. Each combination contains the requesting UE and an SM who is available for assistance and has the video segment requested by that UE. After going through all the active streaming sessions, a list of all possible combinations will be available. The list contains all possible

combinations of requesting UEs and all SMs that are available to assist these requesting UEs. Then, the *CD++2Gurobi* checks whether or not the list is empty. If the list is empty, this obviously means there are currently no SMs to assist. Otherwise, a Gurobi optimization model is built from the available list of combinations. After building the initial Gurobi model, the objective will be set, and the constraints will be imposed on the Gurobi model. This is done using the aforementioned functions provided by the Gurobi C++ library.

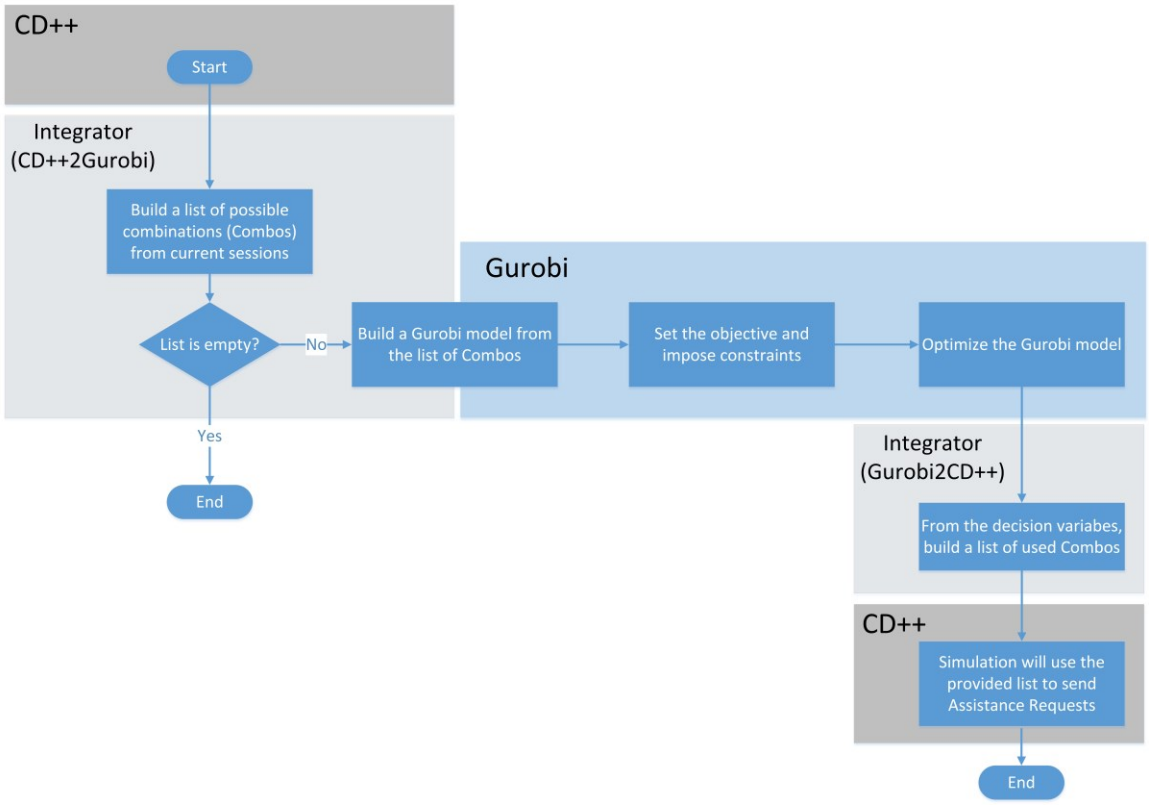

Figure 9.3. Flowchart of the CD++ and Gurobi integration process.

Then, the function *optimize* will be used to optimize the Gurobi model. After the model is optimized, the *Gurobi2CD++* uses the values of the decision variables to return the list of the combinations that are going to be used to the simulation model. This is a list of the SMs that will be used for assistance and the corresponding destination UEs. This will be

used by the BS in the he simulation model to prepare and send assistance request messages to the SMs.

Figure 9.3 shows the component (CD++, *CD++2Gurobi*, etc.) that is involved in implementing each step of the process above. This is depicted from the label of the box that contains that step. For instance, the figure shows that the *CD++2Gurobi* component is responsible for building the list of combinations from the active streaming sessions.

Figure 9.4 shows the UML diagram for the C++ classes built and employed in this integration. These are classed from the DEVS model of the LTE-A network, the integrator, and the Gurobi C++ library. Due to lack of space, we only show the C++ classes that are involved in the integration. For other classes from the DEVS model implementation and the Gurobi C++ library, the reader is referred to Figure 4.2 and Figure 9.2, respectively. As the figure shows, the SM assignment module is implemented with the atomic model *SMAssignMod*. This model is activated whenever the SM assignment is to be performed. An object of this class has, as a variable, an object of the class *Integrator*. As discussed above, the integrator is responsible for building the list of combinations and building the Gurobi optimization model. As such, it has objects of the *GRBEnv* and *GRBModel* classes. It also has a list of objects of the type *Combo*.

The class *Combo* represents a UE-SM combination that could be used. Each combo has an object of the *GRBVar* class. This is a binary decision variable that is set to 0 in the beginning for all *Combo* objects. These same objects of the *GRBVar* will be used in the *GRBModel* object.

After the Gurobi model is optimized, some of the decision variables' values will be set to 1, depending on which ones are selected to maximize the objective function. The UE-SM combinations from the *Combo* objects that has binary decision variables with values of 1 will be returned to the SM assignment module so that corresponding assistance messages are created.

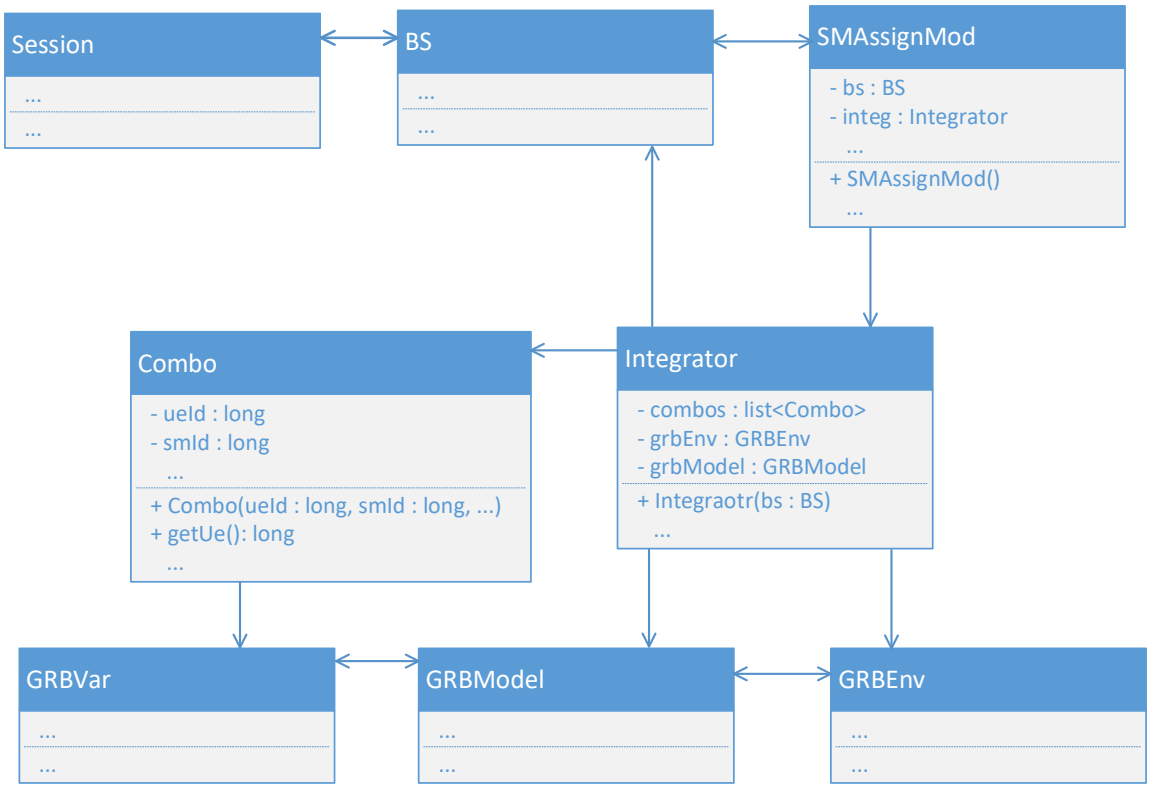

Figure 9.4. UML class diagram of the CD++ and Gurobi integration.

9.4 Performance evaluation

We ran various simulations to evaluate the performance of DABAST in terms of video streaming QoE, with all the techniques discussed above. We use DABAST-CSVD here as our case study, and hence, we drop the name of the caching and distribution algorithm, i.e., we use the name DABAST. In this section, we consider DABAST with Buffer-Based scheduling (BB), DABAST with BB scheduling and High Rate Caching (BB-HRC), and

DABAST with BB scheduling, High Rate Caching, and SM Assignment optimization (BB-HRC-SMA). We measure the same QoE metrics used in the previous chapters. The simulation scenario and setup in the previous chapters (Table 7.1 and Table 7.2) were also used here. As previously discussed in the beginning of this chapter, we consider the case of high rate caching when the traffic load is relatively low (users are not experiencing rebufferings). High rate caching is obviously not recommended in cases where UEs in the cell are experiencing reoccurring video buffer stalling, as it would make the situation worse. As such, we consider a cell with 300 UEs in the simulations. Moreover, a Zipf exponent of 1.5 and 500 videos are considered.

Table 9.1 shows the mean values along with the MoE values for 95% confidence interval of the measured QoE metrics. The table shows the values for the three versions of DABAST, i.e., BB, BB-HRC, and BB-HRC-SMA. The average value for each simulation run was calculated. The values below show the mean of all the average values from 120 simulation runs.

Table 9.1. Simulation results for DABAST (BB vs. BB-HRC vs. BB-HRC-SMA).

	BB		**BB-HRC**		**BB-HRC-SMA**	
	Mean	MoE	Mean	MoE	Mean	MoE
Rebufferings	0.0	0.0	0.0036	0.0004	0.0	0.0
Cont. index	1.0	0.0	0.9997	0.0003	1.0	0.0
Initial delay (sec)	19.280	0.1457	19.580	0.1694	20.222	0.1474
Video bit rate (kbps)	459.06	0.9697	635.10	4.6248	755.00	5.8524

Table 9.1 shows that for BB and BB-HRC-SMA, the average number of rebufferings is 0, while BB-HRC resulted in a negligible increase in the number of rebufferings. Table 9.1

shows that for BB-HRC, the average number of rebufferings is 0.0036. A further inspection of the results of BB-HRC has shown that the maximum number of rebufferings experienced by a UE is 1. This means that out of the 300 UEs, about one UE might experience 1 rebuffering.

Regarding BB, it is expected to have the least average number of rebufferings, as it does not employ high rate caching, which means no video segments are downloaded with high video bit rate. When comparing BB-HRC and BB-HRC-SMA, we can see that in addition to significantly improving the video bit rate, BB-HRC-SMA completely avoided any rebufferings. Thanks to SM assignment optimization, where every time SM assignment is performed, the SMs are assigned in parallel to the UEs, i.e., the optimizer has a global view of the available SMs and the requesting UEs. With BB-HRC, on the other hand, SMs are assigned to UEs sequentially. The assignment module goes through the requesting UEs in a descending order of their scheduling metric (ascending order of their playout buffer length), and each time allocates to that UE the available SM (if any) with the least load, to achieve load balancing between SMs. This way, the SM assignment module might assign to a UE the SM with the least load among the available SMs, although that SM could be the only one available for the next requesting UE. This explains why BB-HRC caused a very slight increase in the average number of rebufferings.

In addition to avoiding rebufferings, BB-HRC-SMA significantly improved the average video bit rate over other versions of DABAST. The results show that BB-HRC achieved 38.35% increase in the average video bit rate over BB. But BB-HRC, as mentioned, caused a very slight increase in the average number of rebufferings. On the other hand,

BB-HRC-SMA achieved 295.94 kbps (64.47%) and 119.9 kbps (18.88%) increase in the average video bit rate over BB and BB-HRC, respectively, without causing any rebufferings.

Regarding the average initial delay, we can see that high rate caching has caused a very slight increase in the average initial delay. The results show that BB-HRC caused only 0.3 second (1.56%) increase in the average delay, and BB-HRC-SMA caused only 0.94 second (4.88%) increase in the initial delay. This is expected as many segments with higher video bit rates are transmitted with high rate caching. However, this is a small price to pay considering the significant improvement achieved in terms of the average video bit rate.

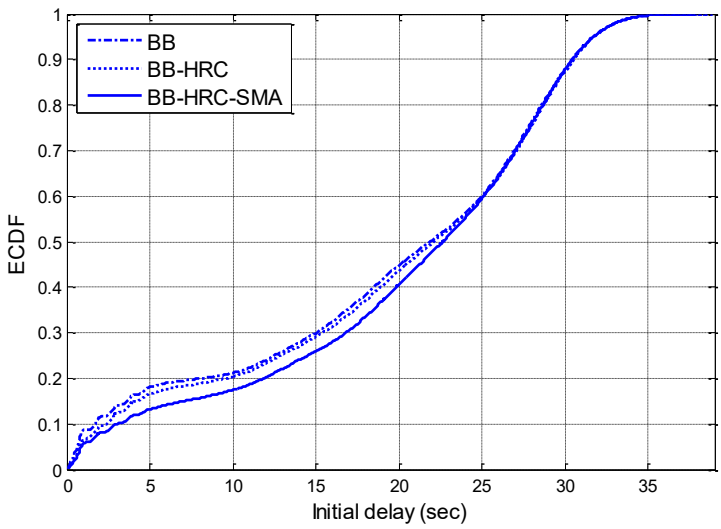

Figure 9.5. ECDF of the initial delay for DABAST (BB vs. BB-HRC vs. BB-HRC-SMA).

Figure 9.5 depicts the ECDF of the initial delay for each version of DABAST, i.e., BB, BB-HRC, and BB-HRC-SMA. From the figure, one can see that all versions of DABAST have very close distribution for the initial delay, with a slight difference in the ECDF of BB-HRC-SMA for values lower than 25 seconds. This means that the slight increase in

initial delay caused by high rate caching is not experienced by a small group of UEs, but rather distributed over the UEs in the cell. Thanks to BB scheduling, where cellular resources are allocated to UEs with low playout buffer, which prevents rebuffering and distributes the extra delay caused by transmission of high video rate segments over all UEs in the cell sharing cellular resources.

We calculated the average maximum initial delay, which is the average maximum delay from all the simulation runs. The average maximum delay values for BB, BB-HRC, and BB-HRC-SMA are 35.31, 35.561, and 35.459, respectively. This means that high rate caching only caused a negligible increase in these values, which further shows the importance of BB scheduling.

Table 9.2 shows the number of video segments received with each video bit rate, for the different versions of DABAST (BB, BB-HRC, and BB-HRC-SMA). When comparing the results of BB and BB-HRC, one can see that with BB-HRC, fewer video segments are received with the lower video bit rates (384 and 768 kbps), and 386315 video segments are received with high video bit rate (2 Mbps). These are the segments of popular videos that are downloaded and cached in the SMs with high video bit rate.

Table 9.2. Count of the received segments with each video bit rate.

	Count		
Video bit rate (kbps)	BB	BB-HRC	BB-HRC-SMA
384	2606646	2360752	2121270
768	633354	492933	491777
2000	0	386315	626953
4000	0	0	0

When comparing the results of BB-HRC and BB-HRC-SMA, one can see that with BB-HRC-SMA, 240638 more video segments are received with video bit rate of 2 Mbps. Most of these 240638 video segments are received with the lowest video bit rate in the case of BB-HRC. This explains the further improvement achieved by BB-HRC-SMA over BB-HRC in terms of the average video bit rate.

9.5 Summary

In this chapter, we consider cases where the available resources are enough to avoid rebufferings. We aim to further increase the utilization of the D2D channel in such cases to improve video streaming QoE for users in the cell by improving their video bit rate. In the first section, we present the first proposed technique; high rate caching. Afterwards, we present the second proposed technique, which is SM assignment optimization. Thereafter, we discuss how CD++ and Gurobi were integrated to provide an environment for modeling, simulation, and optimization. This is implemented to run the proposed SM assignment optimization model during simulations. Finally, we evaluate through simulations the performance achieved with these techniques in terms of video streaming QoE. Results show that high rate caching significantly improves the video bit rate for users in the cell. Furthermore, employing SM assignment optimization further improves the achieved video bit rate while reducing the number of rebufferings, which improves video streaming QoE.

Chapter 10

Conclusion

Video accounted for more than half of the total mobile data traffic in 2016, and it is expected to further increase in the upcoming years. This will escalate the growth of data traffic to be transmitted over cellular networks and raise the challenge for cellular network operators. Furthermore, most of this video traffic is consumed by users via video streaming. Due to the emergence of many platforms for video streaming and Over-The-Top (OTT) media services such as YouTube and Netflix, and due to the improvement of mobile devices, the demand for video streaming over cellular networks has increased rapidly during the last couple of years. Consequently, supporting good Quality of Experience (QoE) video streaming services has become a main concern for cellular network operators. As such, new techniques are much needed to help serving video traffic that is becoming the majority of data traffic over cellular networks. Furthermore, the new techniques should consider the complex, dynamic, and delay-sensitive nature of video streaming traffic to provide end users with good QoE video streaming.

In this thesis, we propose various algorithms and techniques to enhance the delivery of video contents and to improve the QoE of video streaming over cellular networks with high user density. First, we propose two algorithms for progressive caching of video segments in User Equipments (UEs), under high traffic load, and for Peer-to-Peer (P2P) transmission of video contents among UEs using Device-to-Device (D2D)

communication. The algorithms are called *Cached and Segmented Video Download* (CSVD), and *DIStributed, Cached, and Segmented video download* (DISCS). The algorithms send segments of video files to selected UEs in the cellular network (called Storage Members (SMs)), to cache video segments and forward the segments to requesting UEs using D2D communication when needed.

The proposed algorithms do not consider only the transmission of video segments to requesting UEs over D2D links, but also caching of video segments in SMs. CSVD and DISCS progressively cache video segments as requested. Furthermore, Locally-popular videos can be strategically cached to be available to all users in the cell. Although these characteristics are very beneficial in any video streaming scenario with high number of users, they are crucial for high traffic load scenarios where users are requesting recently published contents. A protocol has been defined for both algorithms, including a variety of messages necessary for the communication, and a complete definition of the protocol is described.

We developed analytical models for video caching with CSVD and DISCS and derived analytical closed-from expressions for the hit ratio of both algorithms. Furthermore, we built a suite of models using the Discrete EVent System Specification (DEVS) formalism to model LTE-A cellular networks that employ CSVD and DISCS. System-level simulations were performed to study the performance of CSVD and DISCS in terms of the hit ratio, cell's aggregate data rate, as well as the average data rate. Simulation results show that CSVD and DISCS achieve significant performance improvements in terms of the aggregate and average data rates. Furthermore, the results show that DISCS achieves more improvement over CSVD. This is because DISCS speeds up video caching in the

SMs and improves the hit ratio. This is also because in DISCS, the segments of a popular video file are distributed over multiple SMs. This provides further parallelism when transmitting the video files and further load balancing among SMs, which speeds up the transmission of popular video files.

After developing the algorithms above to improve video contents transmission over cellular networks, we focus on video streaming over cellular networks, as most videos on the web nowadays are accessed via streaming. The parameters of the wireless communication on the Radio Access Network (RAN) between the Base-Station (BS) and UEs have an effect on video streaming QoE. As such, we analyze the impact of the wireless transmission parameters in LTE-A networks on Dynamic Adaptive Streaming over HTTP (DASH)-based video streaming QoE. We consider both cell-level and link-level parameters. We built a model for DASH-based video streaming over an LTE-A network and used the model to study this impact. We run many simulations under various scenarios and provide thorough analysis of the results to evaluate the impact of the considered LTE-A transmission parameters on many QoE metrics. At the cell level, we studied the cell channel bandwidth and the number of UEs in the cell. As both of these parameters control the bandwidth a UE shares, and consequently controls the average data rate, they have a significant impact on all the studied QoE metrics. At the link level, we study the impact of the BS transmission power and the BS-UE distance. Results also show that both of these parameters have a considerable impact on all the studied QoE metrics. In most of the studied scenarios, the average video bit rate was the least affected metric by the studied parameters. This is because in DASH, video bit rate is the last metric it tends to improve. This means that adaptation at the application level masks

some issues at the network and link levels. As such, the exploratory data analysis did not just quantify some intuitive behavior, but also brought some new findings to discus.

Results for the impact of transmission parameters on video streaming QoE show that high traffic load or lack of resources in the cell could cause serous degradation to video streaming QoE in cellular networks. As such, we propose and evaluate an architecture that improves the QoE of video streaming in cellular networks in such scenarios. The proposed architecture employs the following techniques:

- The CSVD and DISCS algorithms presented above, which provide BS-assisted progressive caching of video segments and D2D video transmission in cellular networks. Here, CSVD and DISCS are adopted in the context of video streaming. We evaluate the performance of the proposed architecture in terms of video streaming QoE metrics

- DASH is also employed to allow adaptive video streaming. This is employed in our architecture due to its advantages and to allow support for DASH-based applications

The proposed architecture is called DABAST: **DA**SH-based **B**S-**A**ssisted D2D video **ST**reaming in cellular networks. We use the DEVS formalism to build a model for the proposed architecture in an LTE-A network. Simulations based on this model are used to evaluate the performance of DABAST in terms of video streaming QoE metrics. Results show that the proposed architecture achieves significant improvements in terms of QoE when compared to conventional video streaming over a cellular network, i.e., DASH streaming without D2D/P2P streaming. We also study the impact of the caching and

distribution algorithm (CSVD versus DISCS) on the performance improvements achieved by DABAST. We provide a thorough analysis of the results which brings very interesting findings about the impact of the employed algorithm on video streaming QoE in high user density scenarios.

Although simulation results show that DABAST achieves significant improvements in terms of video streaming QoE metrics for most users in the cell, results show that some users do not receive video streaming service with good QoE, due to repetitive video rebuffering. As such, the operation of DABAST can be further improved by making it aware of video streaming QoE for users in the cell. This can further increase the QoE gains achieved by DABAST by targeting users who are experiencing poor QoE or by improving a certain QoE metric as needed. We employ QoE awareness in three aspects of DABAST, namely, cellular resource allocation, caching of video segments, and SM assignment optimization.

We employ a Buffer-Based scheduling (BB) algorithm in DABAST to introduce QoE awareness to cellular resource allocation. Moreover, we analyze the improvements achieved by this algorithm over state-of-the-art cellular resource allocation algorithms. Results show that the BB algorithm, employed in DABAST, significantly improves video streaming QoE for users in the cell by significantly reducing the number of rebufferings. This is because such approach helps utilizing the scarce cellular resources to serve users with low playout buffer and avoid video rebufferings. Furthermore, employing such approach is more beneficial in DABAST because it increases the utilization of the cached video segments when needed, which helps saving the precious cellular resources for users who obtain video segments exclusively through the BS.

We propose an approach for high video bit rate caching of video segments in DABAST to further improve video bit rate under relatively lower traffic load. Simulation results show that high rate caching helps improving video bit rate for users in the cell. Moreover, we also propose an optimization model for the SM assignment problem in DABAST to dynamically and adaptively maximize the aggregate video bit rate in the cell and minimize the number of rebufferings. We integrate the CD++ toolkit with a commercial optimization solver to run the proposed optimization model in the simulations, and evaluate the improvements achieved by the proposed SM assignment optimization approach. Results show that the proposed SM assignment optimization approach significantly improves the achieved video bit rate without increasing the number of rebufferings in the cell, which improves video streaming QoE.

In future work, the adaptation frequency of video bit rate can be also considered in the operation of DABAST. Some studies have shown that frequent switching could have a negative effect on video streaming QoE [103]. According to [104], it is recommended to keep the frequency of adaptation as small as possible. If video bit rate switching cannot be avoided, its amplitude should be kept as small as possible. According to the above studies, considering the adaptation frequency in the operation of DABAST might have an impact on deciding where to send a video segment from, depending on the available video bit rates in the distributed cache and the current traffic load.

The impact of SMs' power consumption on the performance improvements achieved by DABAST can be studied in future work. An interesting topic for future work is considering the power levels and power consumption of the SMs in the operation of DABAST (caching, distribution, and SM assignment).

Deployment of DABAST in heterogeneous wireless networks is also an intriguing topic to consider for future work. With heterogeneous wireless networks, small low power nodes are overlaid with the macro BS. As discussed in Chapter 3, DABAST can be employed in such environment to enhance the delivery of video segments and improve QoE in areas where small cells' coverage is not available.

Another interesting direction for future work is modeling the impact of the transmission parameters on video streaming QoE metrics. For instance, this can be achieved by using data sets obtained from simulations to build Neural Networks (NNs). This study opens the possibility for using such NNs to estimate some measures regarding the achieved QoE (e.g., worst QoE level) under certain cell-level and link-level transmission parameters. These models/NNs can be used in the network to take proactive actions in order to prevent QoE degradation. For instance, based on the current traffic load in the network (e.g., number of current video streaming users) and available channel bandwidth, the BS can use estimations from the NNs to decide whether to start employing D2D video streaming as per DABAST. This can provide a quick way to estimate the necessity for sending video segments from the distributed cache.

References

[1] ITU, “ICT facts and figures.” [Online]. Available: https://www.itu.int/en/ITU-D/Statistics/Documents/facts/ICTFactsFigures2017.pdf. [Accessed: 20-Jan-2018].

[2] I. F. Akyildiz, W.-Y. Lee, and M. C. Vuran, “A survey on spectrum management in cognitive radio networks,” *IEEE Communications Magazine*, vol. 46, no. 4, pp. 40–48, Apr. 2008.

[3] B. Wang and K. J. R. Liu, “Advances in cognitive radio networks: A survey,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 1, pp. 5–23, Feb. 2011.

[4] Cisco, “Cisco visual networking index: Global mobile data traffic forecast update,” 2017. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html. [Accessed: 28-Feb-2018].

[5] S. Parkvall and D. Astely, “The evolution of LTE towards IMT-Advanced,” *Journal of Communications*, vol. 4, no. 3, pp. 146–154, Apr. 2009.

[6] M. Agiwal, A. Roy, and N. Saxena, “Next generation 5G wireless networks: A comprehensive survey,” *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 1617–1655, Feb. 2016.

[7] G. Fodor *et al.*, “Design aspects of network assisted device-to-device communications,” *IEEE Communications Magazine*, vol. 50, no. 3, pp. 170–177, Mar. 2012.

[8] A. Asadi, Q. Wang, and V. Mancuso, “A survey on device-to-device communication in cellular networks,” *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 1801–1819, Nov. 2014.

[9] B. Kaufman and B. Aazhang, “Cellular networks with an overlaid device to device network,” in *the 42nd Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, USA, 2008, pp. 1537–1541.

[10] K. Doppler, M. Rinne, C. Wijting, C. Ribeiro, and K. Hugl, “Device-to-Device communication as an underlay to LTE-advanced networks,” *IEEE Communications Magazine*, vol. 47, no. 12, pp. 42–49, Dec. 2009.

[11] K. Doppler, M. P. Rinne, P. Janis, C. Ribeiro, and K. Hugl, “Device-to-Device communications; Functional prospects for LTE-advanced networks,” in *IEEE International Conference on Communications Workshops*, Dresden, Germany, 2009, pp. 1–6.

[12] A. Al-Habashna, G. Wainer, G. Boudreau, and R. Casselman, “Improving wireless video transmission in cellular networks using D2D communication,” Provisional patent P47111, 2015.

[13] A. Al-Habashna, G. Wainer, G. Boudreau, and R. Casselman, “Cached and segmented video download for wireless video transmission,” in *Proceedings of the 49th Annual Simulation Symposium*, Pasadena, USA, 2016, pp. 18–25.

[14] A. Al-Habashna, G. Wainer, G. Boudreau, and R. Casselman, “Distributed cached and segmented video download for video transmission in cellular networks,” in *2016 International Symposium on Performance Evaluation of Computer and Telecommunication Systems*, Montreal, Canada, 2016, pp. 473–480.

[15] A. Al-Habashna and G. Wainer, “DEVS-based modeling of cached and segmented video download algorithms in LTE-A cellular networks,” in *20th Communications and Networking Symposium*, Virginia Beach, USA, 2017, pp. 374–385.

[16] A. Al-Habashna and G. Wainer, “Improving video transmission in cellular networks with cached and segmented video download algorithms,” *Mobile Networks and Applications*, vol. 23, no. 3, pp. 543–559, Sep. 2017.

[17] N. Golrezaei, P. Mansourifard, A. F. Molisch, and A. G. Dimakis, “Base-station assisted device-to-device communications for high-throughput wireless video networks,” *IEEE Transactions on Wireless Communications*, vol. 13, no. 7, pp. 3665–3676, Jul. 2014.

[18] C. Kilinc *et al.*, “5G multi-RAT integration evaluations using a common PDCP layer,” in *IEEE 85th Vehicular Technology Conference*, Sydney, Australia, 2017, pp. 1–5.

[19] 3GPP, “Study on new radio access technology: Radio access architecture and interfaces,” Technical report 38.801, Mar. 2016.

[20] B. P. Zeigler, H. Praehofer, and T. G. Kim, *Theory of modeling and simulation: Integrating discrete event and continuous complex dynamic systems*. San Diego: Academic Press, 2000.

[21] G. A. Wainer, *Discrete-event modeling and simulation : A practitioner’s approach*. Boca Raton: CRC Press, 2009.

[22] Cisco, “Cisco visual networking index: Forecast and methodology, 2016–2021,” 2017. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html. [Accessed: 16-May-2018].

[23] Sandvine, “2016 Global internet phenomena: Latin america and north america,” Technical report, 2016.

[24] T. Stockhammer, “Dynamic adaptive streaming over HTTP: Standards and design principles,” in *Proceedings of the 2nd Annual ACM Conference on Multimedia Systems*, New York, USA, 2011, pp. 133–144.

[25] IOS, “Information technology—Dynamic Adaptive Streaming over HTTP (DASH)-Part 1: Media presentation description and segment formats,” ISO/IEC 23009-1:2012, May 2012.

[26] DASH industry forum, “Catalyzing the adoption of MPEG-DASH.” [Online]. Available: http://dashif.org/. [Accessed: 28-Feb-2018].

[27] DASH industry forum, “Guidelines for implementation: DASH-AVC/264 interoperability points,” 2013. [Online]. Available: http://dashif.org/wp-content/uploads/2015/04/DASH-AVC-264-base-v1.03.pdf. [Accessed: 28-Feb-2018].

[28] A. Al-Habashna, S. Fernandes, and G. Wainer, “Analyzing the effect of LTE-A transmission parameters on video streaming quality of experience,” in *Proceedings of the 21st Communications and Networking Symposium*, Baltimore, USA, 2018, pp. 236–247.

[29] V. Chandrasekhar, J. Andrews, and A. Gatherer, “Femtocell networks: a survey,” *IEEE Communications Magazine*, vol. 46, no. 9, pp. 59–67, Sep. 2008.

[30] A. Al-Habashna, S. Fernandes, and G. Wainer, “DASH-based peer-to-peer video streaming in cellular networks,” in *2016 International Symposium on Performance Evaluation of Computer and Telecommunication Systems*, Montreal, Canada, 2016, pp. 481–488.

[31] A. Al-Habashna, G. Wainer, and S. Fernandes, “Improving video streaming over cellular networks with DASH-based device-to-device streaming,” in *2017 International Symposium on Performance Evaluation of Computer and Telecommunication Systems*, Seattle, USA, 2017, pp. 468–475.

[32] A. Al-Habashna and G. Wainer, “DASH-based device-to-device video streaming for cellular networks with high user density,” in *2018 International Symposium on Performance Evaluation of Computer and Telecommunication Systems*, Bordeaux, France, 2018, pp. 1–8.

[33] Y. Li, B. Cao, and C. Wang, “Handover schemes in heterogeneous LTE networks: challenges and opportunities,” *IEEE Wireless Communications*, vol. 23, no. 2, pp. 112–117, Apr. 2016.

[34] Y. Zhang, L. Song, W. Saad, Z. Dawy, and Z. Han, “Contract-based incentive mechanisms for device-to-device communications in cellular networks,” *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 10, pp. 2144–2155, Oct. 2015.

[35] L. Duan, L. Gao, and J. Huang, “Cooperative spectrum sharing: A contract-based approach,” *IEEE Transactions on Mobile Computing*, vol. 13, no. 1, pp. 174–187, Jan. 2014.

[36] L. Duan, T. Kubo, K. Sugiyama, J. Huang, T. Hasegawa, and J. Walrand, “Motivating smartphone collaboration in data acquisition and distributed computing,” *IEEE Transactions on Mobile Computing*, vol. 13, no. 10, pp. 2320–2333, Oct. 2014.

[37] B. Li, Z. Wang, J. Liu, and W. Zhu, “Two decades of internet video streaming,” *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 9, no. 1, pp. 1–20, Oct. 2013.

[38] I. Ullah, G. Doyen, G. Bonnet, and D. Gaiti, “A survey and synthesis of user behavior measurements in P2P streaming systems,” *IEEE Communications Surveys & Tutorials*, vol. 14, no. 3, pp. 734–749, Sep. 2011.

[39] G. Tian and Y. Liu, “Towards agile and smooth video adaptation in dynamic HTTP streaming,” in *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, Nice, France, 2012, pp. 109–120.

[40] L. De Cicco, V. Caldaralo, V. Palmisano, and S. Mascolo, “ELASTIC: A client-side controller for Dynamic Adaptive Streaming over HTTP (DASH),” in *20th International Packet Video Workshop*, San Jose, USA, 2013, pp. 1–8.

[41] T.-Y. Huang *et al.*, “A buffer-based approach to rate adaptation: Evidence from a large video streaming service,” in *Proceedings of the 2014 ACM SIGCOMM*, New York, USA, 2014, pp. 187–198.

[42] L. De Cicco and S. Mascolo, “An adaptive video streaming control system: Modeling, validation, and performance evaluation,” *IEEE/ACM Transactions on Networking*, vol. 22, no. 2, pp. 526–539, Apr. 2014.

[43] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, “A control-theoretic approach for dynamic adaptive video streaming over HTTP,” in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, New York, USA, 2015, pp. 325–338.

[44] Wikipedia, “4K resolution: Video streaming.” [Online]. Available: https://en.wikipedia.org/wiki/4K_resolution#Video_streaming. [Accessed: 05-Jun-2018].

[45] K. Brunnström *et al.*, *Qualinet white paper on definitions of quality of experience*. Lausanne, Switzerland: European Network on Quality of Experience in Multimedia Systems and Services (COST Action IC 1003), 2014.

[46] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hobfeld, and P. Tran-Gia, “A survey on quality of experience of HTTP adaptive streaming,” *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 469–492, 2015.

[47] ITU, “Multimedia quality of service and performance, generic and user-related aspects,” ITU-T P.1080 Rec. Series G, 2008.

[48] ITU, “Vocabulary for performance and quality of service,” ITU-T Recommendation P.10, Amendment 5, 2016.

[49] T. Hoßfeld, M. Seufert, M. Hirth, T. Zinner, P. Tran-Gia, and R. Schatz, “Quantification of YouTube QoE via crowdsourcing,” in *2011 IEEE International Symposium on Multimedia*, Dana Point, USA, 2011, pp. 494–499.

[50] R. K. P. Mok, E. W. W. Chan, X. Luo, and R. K. C. Chang, “Inferring the QoE of HTTP video streaming from user-viewing activities,” in *Proceedings of the 1st ACM SIGCOMM Workshop on Measurements up the Stack*, Toronto, Canada, 2011, pp. 31–36.

[51] T. Hossfeld, S. Egger, R. Schatz, M. Fiedler, K. Masuch, and C. Lorentzen, “Initial delay vs. interruptions: Between the devil and the deep blue sea,” in *4th International Workshop on Quality of Multimedia Experience*, Yarra Valley, Australia, 2012, pp. 1–6.

[52] Y. Qi and M. Dai, “The effect of frame freezing and frame skipping on video quality,” in *2006 International Conference on Intelligent Information Hiding and Multimedia*, Pasadena, USA, 2006, pp. 423–426.

[53] A. Sackl, S. Egger, and R. Schatz, “Where’s the music? comparing the QoE

impact of temporal impairments between music and video streaming," in *5th International Workshop on Quality of Multimedia Experience*, Klagenfurt am Wörthersee, Austria, 2013, pp. 64–69.

[54] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *IEEE Communications Magazine*, vol. 50, no. 7, pp. 26–36, Jul. 2012.

[55] G. Xylomenos *et al.*, "A survey of information-centric networking research," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 2, pp. 1024–1049, Jan. 2014.

[56] S. A. Chellouche, D. Negru, Y. Chen, and M. Sidibe, "Home-box-assisted content delivery network for Internet video-on-demand services," in *2012 IEEE Symposium on Computers and Communications*, Cappadocia, Turkey, 2012, pp. 544–550.

[57] P. Ostovari, J. Wu, and A. Khreishah, "Efficient online collaborative caching in cellular networks with multiple base stations," in *13th IEEE International Conference on Mobile Ad Hoc and Sensor Systems*, Brasilia, Brazil, 2016, pp. 136–144.

[58] S.-H. Shen and A. Akella, "An information-aware QoE-centric mobile video cache," in *Proceedings of the 19th Annual International Conference on Mobile Computing & Networking*, New York, USA, 2013, pp. 401–412.

[59] M. A. Maddah-Ali and U. Niesen, "Decentralized coded caching attains order-optimal memory-rate tradeoff," *IEEE/ACM Transactions on Networking*, vol. 23, no. 4, pp. 1029–1040, Aug. 2015.

[60] J. Zhao, P. Zhang, G. Cao, and C. R. Das, "Cooperative caching in wireless P2P networks: Design, implementation, and evaluation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 2, pp. 229–241, Feb. 2010.

[61] H. J. Kang, K. Y. Park, K. Cho, and C. G. Kang, "Mobile caching policies for Device-to-Device (D2D) content delivery networking," in *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Toronto, Canada, 2014, pp. 299–304.

[62] N. Golrezaei, A. G. Dimakis, and A. F. Molisch, "Wireless device-to-device communications with distributed caching," *arXiv preprint arXiv:1205.7044*, May 2012.

[63] R. K. P. Mok, E. W. W. Chan, and R. K. C. Chang, "Measuring the quality of experience of HTTP video streaming," in *12th IFIP/IEEE International Symposium on Integrated Network Management and Workshops*, Dublin, Ireland, 2011, pp. 485–492.

[64] F. Wamser, M. Seufert, P. Casas, R. Irmer, P. Tran-Gia, and R. Schatz, "YoMoApp: A tool for analyzing QoE of YouTube HTTP adaptive streaming in mobile networks," in *2015 European Conference on Networks and Communications*, Paris, France, 2015, pp. 239–243.

[65] G. Gómez, L. Hortigüela, Q. Pérez, J. Lorca, R. García, and M. C. Aguayo-Torres, "YouTube QoE evaluation tool for Android wireless terminals," *EURASIP Journal*

on Wireless Communications and Networking, vol. 2014, no. 1, p. 164, Dec. 2014.

[66] J. De Vriendt, D. De Vleeschauwer, and D. C. Robinson, "QoE model for video delivered over an LTE network using HTTP adaptive streaming," *Bell Labs Technical Journal*, vol. 18, no. 4, pp. 45–62, Mar. 2014.

[67] P. Casas, P. Fiadino, A. Sackl, and A. D'Alconzo, "YouTube in the move: Understanding the performance of YouTube in cellular networks," in *2014 IFIP Wireless Days*, Rio de Janeiro, Brazil, 2014, pp. 1–6.

[68] P. Casas, R. Schatz, F. Wamser, M. Seufert, and R. Irmer, "Exploring QoE in cellular networks: How much bandwidth do you need for popular smartphone apps?," in *Proceedings of the 5th Workshop on All Things Cellular: Operations Applications and Challenges*, New York, USA, 2015, pp. 13–18.

[69] A. Le, L. Keller, H. Seferoglu, B. Cici, C. Fragouli, and A. Markopoulou, "MicroCast: Cooperative video streaming using cellular and local connections," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2983–2999, Oct. 2016.

[70] A. Detti, B. Ricci, and N. Blefari-Melazzi, "Mobile peer-to-peer video streaming over information-centric networks," *Computer Networks*, vol. 81, pp. 272–288, Apr. 2015.

[71] P. M. Eittenberger, M. Herbst, and U. R. Krieger, "RapidStream: P2P streaming on android," in *2012 19th International Packet Video Workshop*, Munich, Germany, 2012, pp. 125–130.

[72] V. A. Siris and D. Dimopoulos, "Multi-source mobile video streaming with proactive caching and D2D communication," in *IEEE 16th International Symposium on A World of Wireless, Mobile and Multimedia Networks*, Boston, USA, 2015, pp. 1–6.

[73] T. Q. Duong, N.-S. Vo, T.-H. Nguyen, M. Guizani, and L. Shu, "Energy-aware rate and description allocation optimized video streaming for mobile D2D communications," in *2015 IEEE International Conference on Communications*, London, UK, 2015, pp. 6791–6796.

[74] D. Bethanabhotla, G. Caire, and M. J. Neely, "Joint transmission scheduling and congestion control for adaptive streaming in wireless device-to-device networks," in *Proceedings of the 46th Conference on Signals, Systems and Computers*, Pacific Grove, USA, 2012, pp. 1179–1183.

[75] J. Kim, G. Caire, and A. F. Molisch, "Quality-aware streaming and scheduling for device-to-device video delivery," *IEEE/ACM Transactions on Networking*, vol. 24, no. 4, pp. 2319–2331, Aug. 2016.

[76] H. Zhu, Y. Cao, W. Wang, B. Liu, and T. Jiang, "QoE-aware resource allocation for adaptive device-to-device video streaming," *IEEE Network*, vol. 29, no. 6, pp. 6–12, Nov. 2015.

[77] G. Gómez, J. Lorca, R. García, and Q. Pérez, "Towards a QoE-driven resource control in LTE and LTE-A networks," *Journal of Computer Networks and Communications*, vol. 2013, pp. 1–15, Jan. 2013.

[78] A. El Essaili, D. Schroeder, E. Steinbach, D. Staehle, and M. Shehada, “QoE-based rraffic and resource management for adaptive HTTP video delivery in LTE,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 6, pp. 988–1001, Jun. 2015.

[79] S. Cicalo, N. Changuel, V. Tralli, B. Sayadi, F. Faucheux, and S. Kerboeuf, “Improving QoE and fairness in HTTP adaptive streaming over LTE network,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 12, pp. 2284–2298, Dec. 2016.

[80] K. Piamrat, K. D. Singh, A. Ksentini, C. Viho, and J.-M. Bonnin, “QoE-aware scheduling for video-streaming in high speed downlink packet access,” in *2010 IEEE Wireless Communication and Networking Conference*, Sydney, Australia, 2010, pp. 1–6.

[81] T. Ghalut, H. Larijani, and A. Shahrabi, “QoE-aware optimization of video stream downlink scheduling over LTE networks using RNNs and genetic algorithm,” *Procedia Computer Science*, vol. 94, pp. 232–239, Jan. 2016.

[82] G. Lee, H. Kim, Y. Cho, and S.-H. Lee, “QoE-aware scheduling for sigmoid optimization in wireless networks,” *IEEE Communications Letters*, vol. 18, no. 11, pp. 1995–1998, Nov. 2014.

[83] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon, “I tube, you tube, everybody tubes,” in *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, New York, USA, 2007, pp. 1–14.

[84] A. Damnjanovic *et al.*, “A survey on 3GPP heterogeneous networks,” *IEEE Wireless Communications*, vol. 18, no. 3, pp. 10–21, Jun. 2011.

[85] E. Dahlman, S. Parkvall, and J. Sköld, *4G LTE/LTE-advanced for mobile broadband*, Second. Amsterdam: Elsevier, 2014.

[86] 3GPP, “Evolved universal terrestrial radio access; RF system scenarios,” Technical report TR36.942, Dec. 2015.

[87] A. Al-Hourani, S. Chandrasekharan, and S. Kandeepan, “Path loss study for millimeter wave device-to-device communications in urban environment,” in *2014 IEEE International Conference on Communications Workshops*, Sydney, Australia, 2014, pp. 102–107.

[88] R. G. Sargent, “Verification and validation of simulation models,” in *Proceedings of the 2011 Winter Simulation Conference*, Phoenix, USA, 2011, pp. 183–198.

[89] T. H. Naylor and J. M. Finger, “Verification of computer simulation models,” *Management Science*, vol. 14, no. 2, pp. 92–101, Oct. 1967.

[90] J. Banks, J. S. Carson, B. L. Nelson, and D. M. Nicol, *Discrete-event system simulation*, 5th ed. Englewood Cliffs, NJ: Prentice-Hall, 2010.

[91] J. Himmelspach, K. Vanmechelen, and W. Cai, “Implementation and validation of LTE downlink schedulers for ns-3,” in *Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques*, Cannes, France, 2013, pp. 211–218.

[92] A. Abhari and M. Soraya, “Workload generation for YouTube,” *Multimedia Tools and Applications*, vol. 46, no. 1, pp. 91–118, Jan. 2010.

[93] S. Ahsan, V. Singh, and J. Ott, “Characterizing internet video for large-scale active measurements,” *arXiv preprint arXiv:1408.5777v1*, Aug. 2014.

[94] M. S. Ito, D. Bezerra, S. Fernandes, D. Sadok, and G. Szabo, “A fine-tuned control-theoretic approach for dynamic adaptive streaming over HTTP,” in *2015 IEEE Symposium on Computers and Communication*, Larnaca, Cyprus, 2015, pp. 301–308.

[95] ITU-T, “Infrastructure of audiovisual servicescoding of moving video,” ITU-T Recommendation H.264, Jan. 2012.

[96] Nokia, “Quality of Experience (QoE) of mobile services, can it be measured and improved?,” White paper, Nokia Corporation, Finland, 2004.

[97] 3GPP, “Evolved universal terrestrial radio access: Physical channels and modulation,” Technical report TS 36.211, 2015.

[98] T. Kolding, “QoS-aware proportional fair packet scheduling with required activity detection,” in *64th IEEE Vehicular Technology Conference*, Montreal, Canada, 2006, pp. 1–5.

[99] Gurobi Optimization, “Gurobi Optimization - The State-of-the-Art Mathematical Programming Solver,” 2018. [Online]. Available: http://www.gurobi.com/index. [Accessed: 29-May-2018].

[100] Gurobi Optimization, “Mixed-Integer Programming (MIP) basics.” [Online]. Available: http://www.gurobi.com/resources/getting-started/mip-basics. [Accessed: 29-May-2018].

[101] Wikipedia, “Gurobi.” [Online]. Available: https://en.wikipedia.org/wiki/Gurobi. [Accessed: 29-May-2018].

[102] Gurobi optimization, “C++ API details.” [Online]. Available: http://www.gurobi.com/documentation/8.0/refman/cpp_api_details.html. [Accessed: 29-May-2018].

[103] B. Lewcio, B. Belmudez, A. Mehmood, M. Waltermann, and S. Moller, “Video quality in next generation mobile networks — Perception of time-varying transmission,” in *2011 IEEE International Workshop Technical Committee on Communications Quality and Reliability*, Naples, USA, 2011, pp. 1–6.

[104] M. Zink, J. Schmitt, and R. Steinmetz, “Layer-encoded video in scalable adaptive streaming,” *IEEE Transactions on Multimedia*, vol. 7, no. 1, pp. 75–84, Feb. 2005.