Performance Analysis of HTTP Adaptive Video Streaming Services in Mobile Networks

Zakaria Ye

ACADÉMIE D'AIX-MARSEILLE
UNIVERSITÉ D'AVIGNON ET DES PAYS DE VAUCLUSE

THÈSE

présentée à l'Université d'Avignon et des Pays de Vaucluse
pour obtenir le diplôme de DOCTORAT

SPÉCIALITÉ : Informatique

École Doctorale 536 «Agrosciences et Sciences»
Laboratoire d'Informatique (EA 4128)

Analyse de Performance des Services de Vidéo Streaming Adaptatif dans les Réseaux Mobiles

Présentée par Zakaria YE

Soutenue le 02 mai 2017 devant le jury composé de :

M.	Olivier Brun	Chercheur, LAAS-CNRS	Rapporteur
M.	Gerardo Rubino	Directeur de Recherche, IRISA/INRIA	Rapporteur
M.	Tijani Chahed	Professeur, Telecom SudParis	Examinateur
M.	Francesco De Pellegrini	Chercheur, Fondazione Bruno Kessler	Examinateur
M.	Salah Eddine El Ayoubi	Chercheur, Orange Labs	Examinateur
M.	Guillaume Urvoy-Keller	Professeur, Université Nice Sophia Antipolis	Examinateur
M.	Rachid El-Azouzi	Professeur, Université d'Avignon	Directeur de Thèse
Mme.	Tania Jiménez	Chercheur, Université d'Avignon	Co-Directeur de Thèse

Laboratoire d'Informatique d'Avignon - CERI
Centre d'Enseignement et de Recherche en Informatique

ACADÉMIE D'AIX-MARSEILLE
UNIVERSITÉ D'AVIGNON ET DES PAYS DE VAUCLUSE

THESIS

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

In Computer Science

Doctoral School 536 «Agrosciences et Sciences»
Laboratoire d'Informatique (EA 4128)

Performance Analysis of HTTP Adaptive Video Streaming Services in Mobile Networks

Presented by Zakaria YE

Defended on May 02, 2017 before the jury members:

M.	Olivier Brun	Researcher, LAAS-CNRS	Reviewer
M.	Gerardo Rubino	Research Director, IRISA/INRIA	Reviewer
M.	Tijani Chahed	Professor, Telecom SudParis	Examiner
M.	Francesco De Pellegrini	Researcher, Fondazione Bruno Kessler	Examiner
M.	Salah Eddine El Ayoubi	Researcher, Orange Labs	Examiner
M.	Guillaume Urvoy-Keller	Professor, University of Nice Sophia Antipolis	Examiner
M.	Rachid El-Azouzi	Professor, University of Avignon	Advisor
Mrs.	Tania Jiménez	Researcher, University of Avignon	co-Advisor

Laboratoire d'Informatique d'Avignon - CERI
Centre d'Enseignement et de Recherche en Informatique

Résumé

Le trafic vidéo a subit une augmentation fulgurante sur Internet ces dernières années. Pour pallier à cette importante demande de contenu vidéo, la technologie du streaming adaptatif sur HTTP est utilisée. Elle est devenue par ailleurs très populaire car elle a été adoptée par les différents acteurs du domaine de la vidéo streaming. C'est une technologie moins couteuse qui permet aux fournisseurs de contenu, la réutilisation des serveurs web et des caches déjà déployés. En plus, elle est exempt de tout blocage car elle traverse facilement les pare-feux et les translations d'adresses sur Internet.

Dans cette thèse, nous proposons une nouvelle méthode de vidéo streaming adaptatif appelé "Backward-Shifted Coding (BSC)". Il se veut être une solution complémentaire au standard DASH, le streaming adaptatif et dynamique utilisant le protocole HTTP. Nous allons d'abord décrire ce qu'est la technologie BSC qui se base sur le codec (encodeur-décodeur) à multi couches SVC, un algorithme de compression extensible ou évolutif. Nous détaillons aussi l'implémentation de BSC dans un environnement DASH. Ensuite, nous réalisons une évaluation analytique de BSC en utilisant des résultats standards de la théorie des files d'attente. Les résultats de cette analyse mathématique montrent que le protocole BSC permet de réduire considérablement le risque d'interruption de la vidéo pendant la lecture, ce dernier étant très pénalisant pour les utilisateurs. Ces résultats vont nous permettre de concevoir des algorithmes d'adaptation de qualité à la bande passante en vue d'améliorer l'expérience utilisateur. Ces algorithmes permettent d'améliorer la qualité de la vidéo même étant dans un environnement où le débit utilisateur est très instable.

La dernière étape de la thèse consiste à la conception de stratégies de caching pour optimiser la transmission de contenu vidéo utilisant le codec SVC. En effet, dans le réseau, des serveurs de cache sont déployés dans le but de rapprocher le contenu vidéo auprès des utilisateurs pour réduire les délais de transmission et améliorer la qualité de la vidéo. Nous utilisons la programmation linéaire pour obtenir la solution optimale de caching afin de le comparer avec nos algorithmes proposés. Nous montrons que ces algorithmes augmentent la performance du système tout en permettant de décharger les liens de transmission du réseau cœur.

MOTS-CLEFS: Streaming Vidéo, Streaming Adaptatif, Qualité d'Expérience, Backward-Shifted Coding, Scalable Video Coding, Caching.

Abstract

Due to the growth of video traffic over the Internet in recent years, HTTP Adaptive Streaming (HAS) solution becomes the most popular streaming technology because it has been succesfully adopted by the different actors in Internet video ecosystem. It allows the service providers to use traditional stateless web servers and mobile edge caches for streaming videos. Further, it allows users to access media content from behind Firewalls and NATs.

In this thesis we focus on the design of a novel video streaming delivery solution called Backward-Shifted Coding (BSC), a complementary solution to Dynamic Adaptive Streaming over HTTP (DASH), the standard version of HAS. We first describe the Backward-Shifted Coding scheme architecture based on the multi-layer Scalable Video Coding (SVC). We also discuss the implementation of BSC protocol in DASH environment. Then, we perform the analytical evaluation of the Backward-Sihifted Coding using results from queueing theory. The analytical results show that BSC considerably decreases the video playback interruption which is the worst event that users can experience during the video session. Therefore, we design bitrate adaptation algorithms in order to enhance the Quality of Experience (QoE) of the users in DASH/BSC system. The results of the proposed adaptation algorithms show that the flexibility of BSC allows us to improve both the video quality and the variations of the quality during the streaming session.

Finally, we propose new caching policies to be used with video contents encoded using SVC. Indeed, in DASH/BSC system, cache servers are deployed to make contents closed to the users in order to reduce network latency and improve user-perceived experience. We use Linear Programming to obtain optimal static cache composition to compare with the results of our proposed algorithms. We show that these algorithms increase the system overall hit ratio and offload the backhaul links by decreasing the fetched content from the origin web servers.

KEY-WORDS: Video Streaming, Adaptive Streaming, Quality of Experience, Backward-Shifted Coding, Scalable Video Coding, Caching.

Acknowledgements

First, I would like to thank my thesis defense committee especially the reviewers for their helpful comments.

I would like to thank my advisor, Prof. Rachid El-Azouzi. He gave me the opportunity to pursue this PhD thesis and he believed in my capacities to successfully accomplish this adventure. He is a really good teacher who let people thinks themselves. I appreciated that. I am privileged to be his student.

I would also like to thank my co-advisor, Dr. Tania Jiménez. She took care of me since my first day in the LIA. She always encourages me in my daily work and give me many advices. I am very grateful to Tania.

I have been very fortunate to work with Dr. Francesco De Pellegrini. He is a passionate researcher and a fantastic person. I learned a lot from him. He made my stay very amazing in Trento.

I am grateful to many of my friends in Burkina Faso, Morocco, France and elsewhere for their friendship and their support.

Last, but not least, I would like to thank my family for their encouragement and support since the beginning, especially my mother for her daily blessings. I regret that my parents are no longer of this world to attend my dissertation defense, but I know they would be proud of me. This thesis is dedicated to them.

À MA MÈRE... repose en paix

Contents

List of Figures

List of Tables

List of Acronyms

3GPP	**3**rd **G**eneration **P**artnership **P**roject
AVC	**A**dvanced **V**ideo **C**oding
BBA	**B**uffer **B**ased **A**lgorithms
BB-BSC	**B**uffer **B**ased BSC
BL	**B**ase **L**ayer
BSC	**B**ackward **S**hifted **C**oding
CA	**C**arrier **A**ggregation
CDN	**C**ontent **D**elivery **N**etworks
CP	**C**ontent **P**roviders
CoMP	**Co**ordinated **M**ulti **P**oint
CQM	**C**hannel **Q**uality **M**atching
DASH	**D**ynamic **A**daptive **S**treaming over **HTTP**
DRM	**D**igital **R**ights **M**anagement
DSL	**D**igital **S**ubscriber **L**ine
EL	**E**nhancement **L**ayer
EME	**E**ncrypted **M**edia **E**xtensions
EPS	**E**volved **P**acket **S**ystem
FEC	**F**orward **E**rror **C**orrection
FIFO	**F**irst **I**n **F**irst **O**ut
GSM	**G**lobal **S**ystem for **M**obile communications
HAS	**H**TTP **A**daptive **S**treaming
HD	**H**igh **D**efinition
HDS	Adobe **H**TTP **D**ynamic **S**treaming
HEVC	**H**igh **E**fficiency **V**ideo **C**oding
HLS	**H**TTP **L**ive **S**treaming
HSDPA	**H**igh **S**peed **D**ownlink **P**acket **A**ccess
HSPA	**H**igh-**S**peed **P**acket **A**ccess
HTML	**H**yper**T**ext **M**arkup **L**anguage
HTTP	**H**yper**T**ext **T**ransfer **P**rotocol
ICN	**I**nformation **C**entric **N**etworks
IDR	**I**nstantaneous **D**ecoding **R**efresh
IP	**I**nternet **P**rotocol
IRM	**I**ndependent **R**eference **M**odel
ITU	**I**nternational **T**elecommunication **U**nion
JVT	**J**oint **V**ideo **T**eam
KPI	**K**ey **P**erformance **I**ndicator
LFU	**L**east **F**requently **R**equest
LP	**L**inear **P**rogramming
LRU	**L**east **R**ecently **R**equest
LRFU	**L**east **R**ecently **F**requently **R**equest
LTE	**L**ong **T**erm **E**volution
M2M	**M**achine-**t**o-**M**achine
MC	**M**acro **C**ell

MCKP	**M**ultiple **C**hoice **K**napsack **P**roblem
MEC	**M**obile **E**dge **C**aching
MFQ	**M**ost **F**requently **Q**uality
MIMO	**M**ultiple **I**nput **M**ultiple **O**utput
MMFM	**M**arkov **M**odulated **F**luid **M**odel
MNO	**M**obile **N**etwork **O**perators
MOS	**M**ean **O**pinion **S**core
MPD	**M**edia **P**resentation **D**escription
MPEG	**M**oving **P**icture **E**xperts **G**roup
MRQ	**M**ost **R**ecently **Q**uality
MSE	**M**edia **S**ource **E**xtensions
MSS	**M**icrosoft Silverlight **S**mooth **S**treaming
MVC	**M**ultiview **V**ideo **C**oding
NAL	**N**etwork **A**bstraction **L**ayer
NAT	**N**etwork **A**ddress **T**ranslation
ODE	**O**rdinary **D**ifferential **E**quations
OTT	**O**ver-**T**he-**T**op
PDE	**P**artial **D**ifferential **E**quations
PF	**P**roportional **F**air
QoE	**Q**uality **o**f **E**xperience
QoS	**Q**uality **o**f **S**ervice
RA	**R**ate **A**ware
RAN	**R**adio **A**ccess **N**etwork
RN	**R**elay **N**ode
RR	**R**ound **R**obin
RSVP	**R**esource **R**eser**V**ation **P**rotocol
RTCP	**R**eal **T**ime **C**ontrol **P**rotocol
RTP	**R**eal-time **T**ransport **P**rotocol
RTSP	**R**eal **T**ime **S**treaming **P**rotocol
RTV	**R**eal-**T**ime **V**ideo
SAP	**S**tream **A**ccess **P**oints
SC	**S**mall **C**ell
SNR	**S**ignal to **N**oise **R**atio
SVC	**S**calable **V**ideo **C**oding
TBA	**T**rhoughput **B**ased **A**lgorithms
TB-BSC	**T**hroughput **B**ased BSC
TCP	**T**ransmission **C**ontrol **P**rotocol
UDP	**U**ser **D**atagram **P**rotocol
UE	**U**ser **E**quipment
UHDTV	**U**ltra **H**igh **D**efinition **T**ele**V**ision
UMTS	**U**niversal **M**obile **T**elecommunications **S**ystem
URL	**U**niform **R**esource **L**ocator
VBR	**V**ariable **B**it **R**ate
VCEG	**V**ideo **C**oding **E**xperts **G**roup
VCL	**V**ideo **C**oding **L**ayer
VoD	**V**ideo **o**n **D**emand
WCDMA	**W**ideband **C**ode **D**ivision **M**ultiple **A**ccess
XML	**E**xtensible **M**arkup **L**anguage

List of Symbols

N	video file size (in frames)
K	number of segments in a video file
ϕ	offset between BL and EL
λ	Poisson arrival rate
μ	Poisson service rate
ρ	traffic load
x	startup threshold (in frames)
T_x	startup delay (in seconds)
N_A	number of votes of candidate A
N_B	number of votes of candidate B
P_s	probability of buffer starvation
$P_s^<$	probability of starvation for $\phi \leq x$
$P_s^>$	probability of starvation for $\phi > x$
$P_s(1)$	probability of having one starvation
$P_s(j)$	probability of having j starvations
$G(z)$	probability generating function of starvation events
Q	infinitesimal generator of the Markov chain
$\mathbf{q}$	stationary regime of the Markov chain
$E[S_i]$	expected time spent in state i before reaching the absorbing state
$E[\tau_x]$	expected time to absorption starting from initial state x
$b_{\mathcal{L}}$	low bitrate
$b_{\mathcal{H}}$	high bitrate
$T_{\mathcal{L}}$	time spent in low bitrate
$T_{\mathcal{H}}$	time spent in high bitrate
b_{avg}	average bitrate
$\mathcal{R}$	set of available bitrates
R_{min}	lowest bitrate in the set $\mathcal{R}$
R_{max}	highest bitrate in the set $\mathcal{R}$
$R_{k,B}$	bitrate of the low layer segment in block k
$R_{k,E}$	bitrate of the top layer segment (with its low layer) in block k
T_s	duration of a segment in seconds
$d_{k,B}$	size of the low layer segment in block k
$d_{k,E}$	size of the top layer segment (without its low layer layer) in block k
$\hat{A}_t$	estimated thoughput at time t after the download of segment $k-1$
$\mathcal{B}_k$	buffer occupancy in seconds of video content when receiving segment k
ϕ_t	duration of the offset in seconds of video content
R_{t-}	highest available bitrate regarding $\hat{A}_t$
R_{t+}	smallest available bitrate regarding $\hat{A}_t$
r	reservoir level in seconds
c	cushion level in seconds
F	adjustment function
d	video playback frequency in frames per second
U	number of users in the macro cells and the small cells

M	cache capacity in bytes
I	total number of video files in the origin server
L	number of quality levels for a video
T_i	duration of video i in seconds
$\mathcal{C}$	set of channel capacities
C_k	channel state corresponding to rate R_k
$x_{i,l}$	indicating variable encoding the event: content i with quality l
π_k	ergodic stationary probability that the user channel is in state R_k
n_k	number of UEs with channel state C_k
ω_i	aggregate demand rate for video i
Λ	objective function
Q_{req}	last requested quality
Q_{freq}	most requested quality
V	total number of videos in the cache
v_l	number of videos with quality l in the cache
s_{il}	size in bytes of video i with quality l
α	Zipf's distribution exponent

Chapter 1

Introduction

In this chapter, we first present the context and the objectives of the thesis. We discuss recent evolutions in telecommunications networks: 4G and 5G networks. We show that despite the important data rate that offer those networks, it is still challenging to properly deliver video content. We give a brief historic of video streaming technologies. Then, we introduce the main contributions of the thesis and finally list the publications at the end of the chapter.

1.1 Context

Despite the evolutions in telecommunication networks, video streaming delivery over wireless networks is still challenging because of the increasing of video demand. This tremendous video traffic demand is mostly due to the pervasive streaming mobile service generated by over-the-top video content providers. According to Cisco visual networking index foreceast [1], the monthly global mobile data traffic will be 30.6 exabytes by 2020. For example, as shown in Fig. 1.1, mobile video will generate three-quarters of mobile data traffic by 2020.

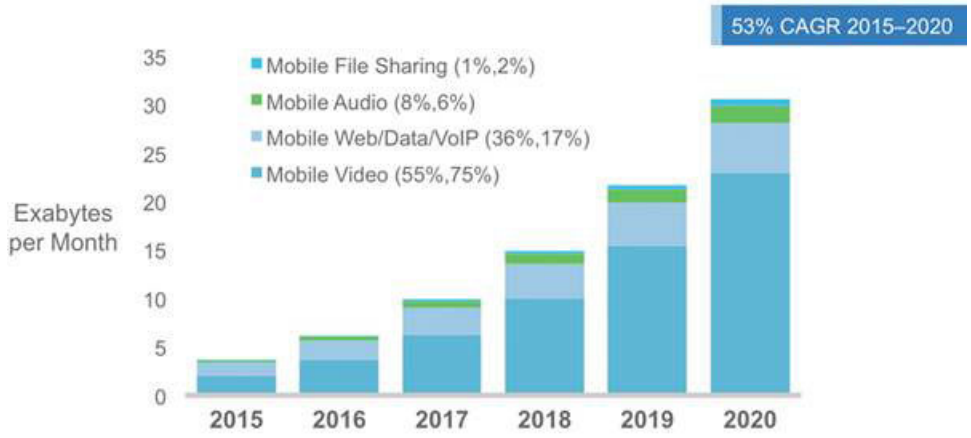

FIGURE 1.1: Cisco VNI mobile, 2016, percents in parentheses refer to 2015 and 2020 traffic share.

In this section, we discuss recent evolutions in telecommunication networks and video streaming technologies.

1.1.1 Recent evolutions in Telecommunication networks

The 3^{rd} Generation Partnership Project (3GPP) has introduced the Long Term Evolution (LTE) of Universal Mobile Telecommunications System (UMTS) known as

4G network[1]. It is the next step forward in cellular 3G services. LTE offers significant improvements over previous technologies such as Global System for Mobile communications (GSM), Universal Mobile Telecommunications System (UMTS) and High-Speed Packet Access (HSPA) by reforming the core network and introducing a novel physical layer. The LTE performance requirements include a high data rate (1Gbps) and spectrum efficiency for both downlink and uplink transmission (30 bits/s/hz in downlink), a low latency ($<$ 100 ms in control plane and $<$ 10 ms in user plane), a high cell capacity than the currently deployed technologies [2]. LTE is an all-IP network. Its architecture is simpler and contains the core network and the Radio Access Network (RAN) which consists of a set of eNodeBs (Bases Station) and User Equipments (UEs) communicating through the air interface. Further, there is the interoperability between LTE networks and W-CDMA, GSM systems and non-3GPP systems.

LTE technology is widely adopted and deployed. According to Cisco, 4G worldwide traffic is more than half of the total mobile traffic in 2016. Further, 4G connections will have the highest share (40.5%) of total mobile connections by 2020. 4G technology full requirements, according to report ITU-R M2134 [5], can be obtained in LTE-Advanced by the use of techniques such as carrier aggregation (CA), Multiple Input Multiple Output (MIMO) 8x8, support of Relay Node (RN), Coordinated Multi Point (CoMP), etc.

The new 5G technology is under standardization. There are two visions for the 5G technology. The hyper-connected vision which suggests that mobile operators will create a combination of existing technologies covering 2G, 3G, 4G, Wi-Fi and others in order to allow higher network density in terms of cells and devices. The next-generation radio access technology vision with a set of requirements to reach such as in the case of 4G technology. The two approaches will be grouped in 5G. The technology requirements includes higher speeds ($>$ 1Gbps) and capacity, much lower latency ($<$ 1 ms), high number of connected devices, 100% coverage, 90% reduction in network energy usage, etc. 5G will offer services such as virtual reality, augmented reality, immersive or tactile Internet, autonomous driving, connected cars, wireless cloud-based office, multi-person videoconferencing, machine-to-machine connectivity (M2M), etc.

Despite the efforts made to increase the performance of the networks, users are still more and more unsatisfied of their received video quality, especially those at cells edge. Many reasons are behind that. First, users expectations of video quality are steadily rising, i.e., they are progressively willing for High Definition (HD) and ultra high definition (4K and 8K resolutions) content. Further, the total number of smart devices is increasing (Fig. 1.2), e.g., those mobile and non-mobile devices capable to display multimedia content. Therefore, it is still challenging to deliver good video quality. In the next section, we review video streaming technologies.

1.1.2 Video streaming technologies

The key concept of streaming is to start processing the media data (audio or video) before it is completely received. Streaming is opposed to downloading where the end user should wait to obtain the entire media file before watching or listening to it. Streaming can be broadly divided into on-demand and real-time categories. With on-demand streaming the video file is stored on a server and the clients request it. With real-time streaming the content is generated and transmitted on the fly with a slight and consistent delay. Real-time streaming can be point-to-point or broadcast. Both on-demand and real-time manage a buffer to store the received data before it is displayed

[1] 4G also includes Worldwide Interoperability for Microwave Access (WiMAX) network technologies.

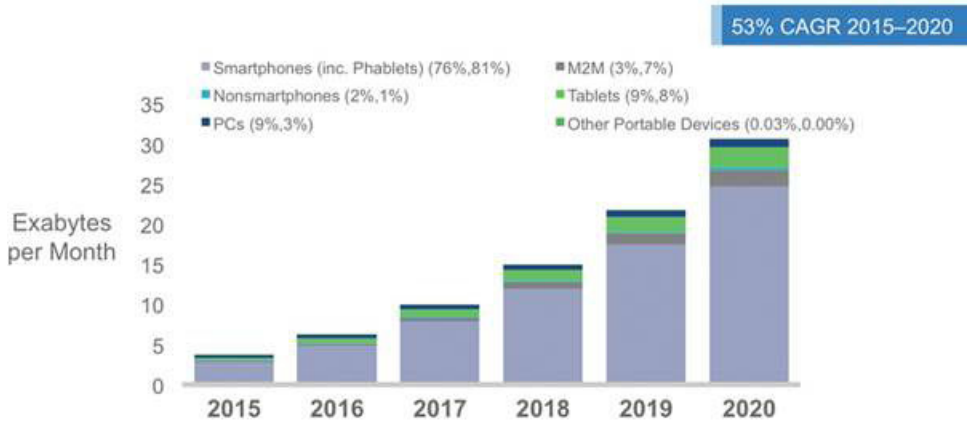

FIGURE 1.2: Cisco VNI mobile, 2016, percents in parentheses refer to 2015 and 2020 device share.

at the screen. The buffer is very important in streaming since it allows to cope with the delay caused by the data transmission through the network. We consider on-demand streaming in this thesis.

Streaming involves protocols at different layers such as the transport, session, presentation and application layer. A video streaming technology may be defined by the used protocols and the way it streams the media. At the transport layer, the two protocols used are Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). TCP is a reliable transmission protocol, it proceeds to the retransmission of lost packets. UDP is lightweight compared to TCP and will keep delivering data rather than put extra effort into re-sending lost packets. However, firewalls may block UDP protocol and only allow TCP.

Real-time Transport Protocol (RTP)

The Real-time Transport Protocol (RTP) is a transport protocol build on UDP designed specifically for real-time transferts [6]. RTP is associated with the Real Time Control Protocol (RTCP) to monitor the quality of service and to convey information about the participants in an on-going session. It provides feedback on the quality of the transmitted data, hence allowing actions such as adjusting the data rate. The Real Time Streaming Protocol (RTSP) is a presentation layer protocol used with RTP. It is similar to Hyper Text Transfer Protocol (HTTP), it carries the clients requests and initiate activities such as playing, pausing and recording. The Resource Reservation Protocol (RSVP) is used to set up the sessions. RTP is used by Flash, RealPlayer, Windows Media Player and QuickTime player.

Progressive downloading

Progressive downloading [7] does not use specific protocols. It follows the streaming concept, that means, the users can start watching the video before it is completely downloaded. The service let the users download the video file as fast as possible regarding the network bandwidth. An initial delay is introduced at the beginning of the video session in order to accumulate a certain amount of video data that prevents from the buffer starvation (empty buffer). The buffer starvation can however happen because the quality of the video is not adapted to the network bandwidth.

HTTP Adaptive Streaming (HAS)

HTTP adaptive streaming (HAS) uses HTTP protocol over TCP [8]. It is the new way of delivering video content that has raised the interest of market in recent years. It uses HTTP for efficient bypassing of NATs and firewalls. Further, TCP packets retransmission delay can be absorbed by buffer deployment at the cliend side. HAS continuously adapts the video quality to the network bandwidth by increasing and decreasing the quality per segment based. Therefore, the video should be available in multiple copies called representations, each representation represents a given quality.

An encoding process is performed on the original video source to provide the different representations. The hardware/software used to encode the video is called codec. The efficiency of nowadays codecs is a huge advance in video streaming technology since it allows us to deliver high video quality with relatively low cost bandwidth. In this thesis, we use Scalable Video Coding (SVC) codec to design a novel video transmission scheme. More details about codecs, especially SVC codec, are provided in the next chapter.

The standard version of HAS is Dynamic Adaptive Streaming over HTTP (DASH) released by the Moving Picture Expert Group (MPEG). HTTP adaptive streaming technology is quite simple and has the advantage to reuse the existing and cost-effective HTTP-based Content Delivery Networks (CDN), proxies and caches.

1.2 Objectives

The objective of this thesis is to improve the performance of video streaming delivery over mobile networks. For this purpose, we use two approaches: a novel and complementary solution to Dynamic Adaptive Streaming over HTTP (DASH) and video caching in DASH context using Scalable Video Coding (SVC).

In Dynamic adaptive streaming over HTTP standard, the way the video quality is adapted to the network bandwidth is not standardized. Therefore, there is no single way to do the quality adaptation. Our proposed adaptive video streaming solution is called Backward-Shifted Coding (BSC). It is based on SVC codec. This novel video delivery scheme is a complementary solution to Dynamic Adaptive Streaming over HTTP standard rather than a simple quality adaptation algorithm. This scheme is cost-effective since it does not require modifications on DASH architecture. We give design details of BSC scheme and perform an analytical evaluation using QoE cost function which incorporates QoE metrics such as the probability of the video playback interruption (buffer starvation), the number of playback interruption, the initial delay (startup delay) and the average video quality. Finally, we propose video quality adaptation algorithms to be used in BSC system.

Caching is a technique used in video streaming systems. By caching contents close to the end users, we reduce transmission delays and improve the user experience. However, the paradigm according to which a video is a unique content does not hold in the case of DASH standard. Thus, relevant caching policies are necessary to optimize the caches composition by taking into account the quality of the cached videos. We propose such rate-aware caching algorithms in the second part of the thesis.

1.3 Contributions

In this section, we bring out the contributions of this thesis. We first show our works about video streaming analysis and QoE modeling, respectively, in **[C1]** and **[C2]** which are not included in the thesis. Thereafter, we cite the contribution of each chapter.

We evaluate the video streaming traffic over the LTE networks in **[C1]**. We use the Vienna LTE system level simulator [4] which simulates network functionalities such as scheduling, mobility handling, interference management and signals propagation. We consider the video streaming service and analyse the video frames interarrival times of the users. The transmission channel is under the effects of the fading (slow and fast fading) and the interference of other users and cells. We consider several standard probability density functions and fit the traffic distribution to those functions as in Fig. 1.3. The results show the efficiency of LTE networks for video streaming service since the

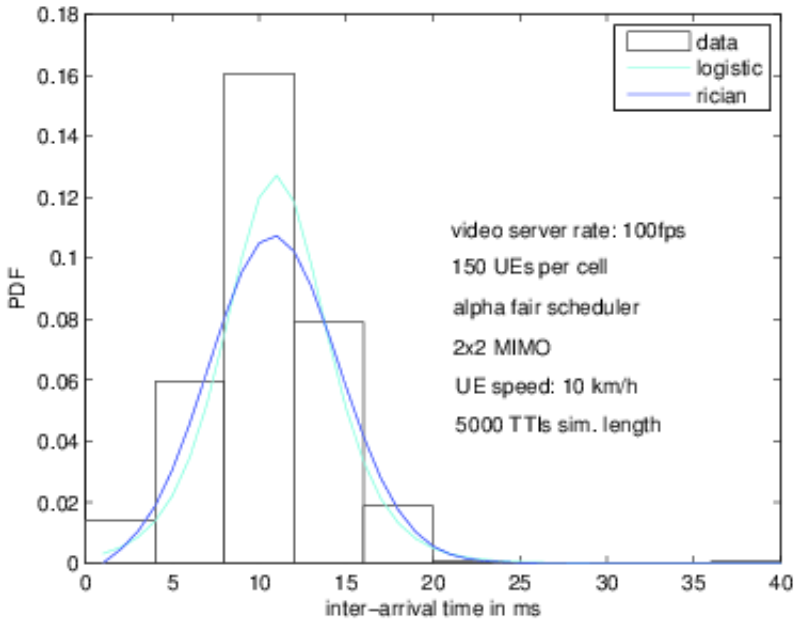

FIGURE 1.3: Probability density function of the video frames interarrival times.

most of the interarrival times follow in the interval 10ms-20ms. For instance the mean service time of the video playback is 40ms for a playback frequency of 25 frame per second.

We investigate QoE metrics computation using fluid models in **[C2]** without any constraint in the video file size. The file size is N in number of frames. The network is modeled as a Markov Modulated Fluid Model (MMFM). We derive partial differential equations and we use the double Laplace Transform method to solve the equations. The double Laplace Transform (LT) method is used in [35] to reduce the PDEs that govern the system to an eigenvalue problem of a matrix equation in the Laplace Transform domain. We use in **[C2]**, a polynomial method using results of first passage time distribution of [36]. A first LT on the PDE gives an ODE. This ODE is solved using the determinant of the equation as a polynomial to obtain LT of the solution. Then, an inverse LT is performed on the LT solution using inversion methods of [37, 38].

In the following, we cite the contribution of each chapter:
In **Chapter 2**, we present the related works on HTTP adaptive video streaming and video caching. We begin by introducing the video streaming system architecture that we study in the thesis. We also review video codecs used in video streaming system

to compress the raw video source before transmission. Then we relate the notions about the Quality of Experience (QoE) since we use it to evaluate the performances of our proposed Backward-Shifted Coding (BSC) system afterwards. We also show existing adaptions methods proposed to enhance the performance of HTTP adaptive video streaming. Finally, we review existing caching solutions and show that these caching policies fail to give plain satisfaction in the context of adaptive video streaming.

In **Chapter 3**, we explain the Backward-Shifted Coding (BSC) scheme. Since BSC is based on existing multi-layers video codecs and in particular the Scalable Video Coding, we first describe this video compression algorithm in details. Then, we give the basic idea of the Backward-Shifted Coding and how we can incorporate it in Dynamic Adaptive Streaming over HTTP (DASH), which becomes nowadays de facto video transmission standard. We finally discuss about the implementation of BSC protocol in DASH. The contributions of this chapter are found in **[C3]** and **[W6]**.

In **Chapter 4**, we analyse the performance of the Backward-Shifted Coding scheme. We adopt a QoE-based analysis since the quality of experience is more suitable for analysing operational efficiency of complex systems such as video streaming systems. In our analysis we compute the probability of the buffer starvation and the probability generating function of the starvation events by using the Ballot theorem. We next compute the startup delay and the average video quality using the quasi-stationary distribution of birth-death processes. Finally, we propose a QoE objective function using the prvious computed metrics in order to evaluate BSC scheme. The results about BSC performances analysis are published in **[C3]**.

In **Chapter 5**, we study bitrate adaptation algorithms based on the Backward-Shifted Coding (BSC) scheme. We perform the integration of BSC in Dynamic Adaptive Streaming over HTTP. As shown in DASH, the video sequence is divided in smaller chunks called segments, each one is provided in multiple qualities. BSC consists on transmitting two superposed segments: a low layer segment and a top layer segment. The main contribution in this chapter is the selection of the quality (bitrate) of the low layer segment and the top layer segment given the network capacity in order to minimize the video playback interruptions, increase the video quality and decrease the video quality variations. Then, we propose bitrate adaptation algorithms in BSC system able to balance the need for maximum quality, video rate smoothness while avoiding the risk of playback interruptions. Finally, we provide simulations using synthetic and real-world video traffic traces to show the benefits of the Backward-Shifted Coding system compared to dynamic adaptive streaming over HTTP standard. The contributions of the bitrate adaptation in Backward-Shifted Coding are published in **[C4]**.

In **Chapter 6**, we study video caching problem in HTTP adaptive streaming systems. Indeed, a solution to the ever increasing video traffic is Mobile Edge Caching (MEC). Caching videos close to eNodeBs (eNBs) and small cells (SCs) relieves congestion at the network operators' backhaul and enables also better control of the quality of experience (QoE) of streamed videos. With adaptive video streaming, each video is provided in multiple copies. In this chapter, we study schemes for mobile edge caching which optimize the stored playout video rate of cached copies. We propose new Rate-Aware (RA) caching policies and determine the static optimal caching strategy by using a Linear Programming (LP) formulation. This provides a reference bound for the performance of the proposed caching algorithms. Finally, we show through numerical simulations that the proposed caching algorithms enhance the system overall performances compared to

existing caching policies. The results of video caching in HTTP adaptive streaming are found in **[S7]**.

In **Chapter 7**, we summarize the results of the previous chapters about the performance analysis of BSC system, the bitrate adaptation algorithms in BSC system and the video caching in HTTP adaptive streaming using the scalable video coding. Then, we discuss about the future works of the thesis. These future works include the quality of experience management, HTTP adaptive video streaming and video caching.

1.4 Publications

The list of publications including international conferences and workshops are following.

International Conferences:

[C1]: **Zakaria Ye**, Tania Jiménez and Rachid El-Azouzi."Video streaming analysis in Vienna LTE system level simulator." *Proceedings of the 8th International Conference on Simulation Tools and Techniques*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), August 2015, Athens, Greece.

[C2]: **Zakaria Ye**, Rachid EL-Azouzi and Tania Jiménez. "Analysis and modelling Quality of Experience of video streaming under time-varying bandwidth." *Wireless and Mobile Networking Conference (WMNC)*, 2016 9th IFIP. IEEE, July 2016, Colmar, France.

[C3]: **Zakaria Ye**, Rachid EL-Azouzi, Tania Jiménez, Eitan Altman and Stefan Valentin. "Backward-Shifted Strategies Based on SVC for HTTP Adaptive Video Streaming." *Proceedings of the 15th International IFIP Networking Conference*, IFIP, May 2016, Vienna, Austria.

[C4]: **Zakaria Ye**, Rachid EL-Azouzi, Tania Jiménez, Francesco De Pellegrini and Stefan Valentin. "Bitrate Adaptation in Backward-Shifted Coding for HTTP Adaptive Video Streaming." *In IEEE International Conference on Communications (ICC)*, May 2017, Paris, France.

Workshops:

[W5]: **Zakaria Ye**, Rachid EL-Azouzi, Tania Jiménez and Eitan Altman. "A Queueing Theoretic Approach to Design a Backward-Shifted Strategies Based on AVC/SVC for HTTP Adaptive Streaming." Poster Presentation, *1st GdR MaDICS Workshop on Big Data for the 5G RAN*, November 2015, Paris, France.

[W6]: **Zakaria Ye**, Rachid EL-Azouzi, Tania Jiménez and Stefan Valentin. "Backward-Shifted Coding (BSC) for HTTP Adaptive Streaming." *11ème Atelier en Evaluation de Performances*, du 15 au 17 mars 2016, Toulouse, France.

Submitted Papers:

[S7]: **Zakaria Ye**, Francesco De Pellegrini, Rachid EL-Azouzi and Tania Jiménez. "Rate Aware Video Caching Schemes for 5G Networks." *Submitted to ITC*, September 2017, Genova, Italy.

Chapter 2

Background

In this chapter, we present a survey of the works on HTTP adaptive video streaming and video caching. We give an overview of Dynamic Adaptive Streaming over HTTP standard by presenting the typical video streaming system architecture, the video codecs and the Quality of Experience (QoE). We see how the quality of experience is used to analyse the performance of video streaming systems. Then, we review existing adaption methods proposed to enhance the performance of HTTP adaptive video streaming. Finally, we relate existing caching solutions and show that these caching policies fail to give satisfaction in the context of adaptive video streaming.

2.1 Video streaming system architecture

2.1.1 Video streaming system

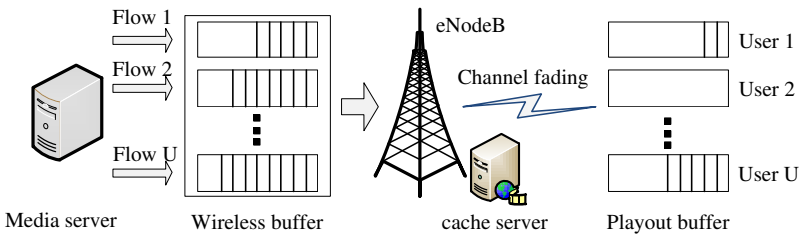

FIGURE 2.1: A typical Video streaming system with U users

We present a typical video transmission system in Fig. 2.1. The system contains the three (3) main components involved in video stream transmission: The media server, the Base Station (BS) (equivalently the network) with local cache and the streaming users (U users in the figure). The mobile users request streaming service from the video server and the media streams are transmitted through the network over a fast fading channel. The streaming flow spans a wired link between the media server and the base station. In this link the streams are transmitted in a regular manner. When the requested videos are within the local cache, there is no need to retrieve them from the distant media server. They are served directly reducing the network latency. Then, the streams span a wireless link, between the base station and the mobile users, which represents the bottleneck of the transmission depending on the channel conditions. Each mobile user is associated with only one flow.

The system is composed of two types of buffers, the wireless buffer at the base station and the playout buffer at the user side. The wireless buffer and the playout buffer work in a tandem way, which means that the flow departure process of a user

in the former is exactly the flow arrival process in the latter. Though, they work at different time scales and at different layers. The wireless buffer works at the bit level due to the bit loading in the lower layers, and the playout buffer works at the video frames level.

Uncompressed video content requires a huge amount of resources for the transmission [9]. For example, with a resolution of 640x480 and a frame size of 300Kb (low quality), one minute video size is bigger than 1Gbytes. Therefore, we need to reduce the size of the video before the transmission through the network, this process is called video coding or video compression. There is the encoding and the decoding process. The software/hardware responsible of the encoding process is the codec. The efficiency of the video transmission strongly depends on the efficiency of the video codec. We review the video codecs in the next section.

2.1.2 video codecs

A non exhaustive list of popular video codecs that are currently used is: H264 Advanced Video Coding (H264/AVC), H264 Scalable Video Coding (H264/SVC), H264 Multiview Video Coding (H264/MVC), Real-Time Video (RTV), H265 High-Efficiency Video Coding (H265/HEVC), VP8. A codec can be determined by its coding efficiency, the complexity of the encoding and decoding processes, the robustness to data losses and errors, the end-to-end delay.

H264/AVC [10] is currently the most commonly used format for the recording, compression and distribution of high definition videos since its first version in May 2003. AVC results in the evolution of ITU-T prior video coding standards summarized in Table 2.1. It is jointly developped by the ITU-T Video Coding Experts Group (VCEG)

name	**date**	**major features**
H261	1990	ISDN video conferencing
H263	1996	based on H261 with increased coding performance
MPEG-1	1991	digital storage media, CD-ROM, at bit rates up to 1.5Mbps
MPEG-2 (H262)	1994	higher data rates for HDTV, SDTV, DVD
MPEG-4 Part 2	2000	mobile applications and streaming
H264 (MPEG-4 Part 10 Advanced Video Coding)	2005	broadcasting television, high definition DVD, digital storage, mobile applications

Table 2.1: Evolution of ITU-T codecs standards

and the ISO/IEC JTC1 Moving Picture Experts Group (MPEG). The project partnership effort is known as the Joint Video Team (JVT). H264/AVC is a single layer codec, i.e., it renders the video with a single bitrate (quality). The algorithm is applied on the video images to render video frames. AVC is used on YouTube, iTunes store, Adobe Flash Player, Microsoft Silverlight, etc. The H264/AVC codec specification includes 21 different extensions (called profiles) among which the Multiview Video Coding (MVC) [11] and the Scalable Video Coding (SVC) [12, 13] which is detailed in Section 3.1.

Besides, H264/AVC and its extensions, there is the new video codec H265/HEVC [14] (and its extensions) which is supposed to replace the family of AVC codec in the next years. The first version was introduced in January 2013. H265/HEVC will deliver the

same video quality as H264/AVC at up to a 50 percent savings in bandwidth. Further, it will provide support for ultra-high definition video such as 4K UHDTV (2160p) and 8K UHDTV (4320p). But it will also require significant processing power to attain its full potential.

Real-Time Video codec is Microsoft's proprietary solution which includes forward error correction (FEC) algorithms in order to control errors in data transmission over noisy communication networks. VP8 [15] is an open-source codec created by On2 Technologies and it is used in Skype for video calls.

2.2 Quality of Experience in video streaming

2.2.1 Definition

The notion of Quality of Experience (QoE) has emerged since the late 90's. It has gained a particular attention because the de facto notion of quality, e.g., the Quality of Service (QoS), was limited to fully express the performance of modern communication systems which are very complex. The Quality of Service is expressed by network parameters like bandwidth, delay, jitter or packet loss. However, a good QoS does not garantee that all customers experience the service to be good. Whereas QoS represents a network-centric view of the performance of the system, the Quality of Experience represents the user-centric view (Fig. 2.2). Therefore, the Quality of Experience expresses

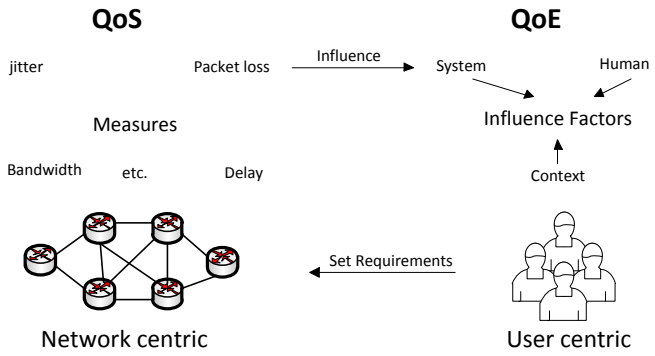

FIGURE 2.2: Conceptual difference of QoS and QoE [16]

directly the user satisfaction of a service. It is influenced by the system, the users and the context.

The first introduction of the term QoE was predominantly promoted by the industry. In 2005, Nokia introduced this concept as a perception of the end users about a service quality [17] "QoE is how a user perceives the usability of a service when in use - how satisfied he or she is with a service". Recently, efforts from the scientific community have been made to agree on a working definition of QoE. The European Network on Quality of Experience in Multimedia Systems and Services, Qualinet [18] in collaboration with the participants of the Dagstuhl Seminar 12181 [19] defines the Quality of Experience as "the degree of delight or annoyance of a user of an application or service. It results from the fulfillment of his or her expectations with respect to the utility and / or enjoyment of the application or service in the light of the user's personnality and current state". Thus, QoE is a subjective concept that depends on the users.

For networks and services dimensioning and control purpose, this qualitative user perception needs to be translated into quantitative input. This quantitative input can be used by the video streaming ecosystem – content providers, content delivery network, analytics services, video player designers and users – at improving QoE prediction and control.

2.2.2 QoE metrics

To quantify the users' perception of the quality of video streaming applications, QoE metrics have been defined and used by industry and scientific comunity. They are the QoE Key Performance Indicators (KPI). A QoE KPI is a user-based metric which can capture the user' perception of a service. The following metrics are used in adaptive video streaming:

- **Initial delay or Startup delay:** it denotes the duration (measured in seconds) between the time that a user initiates a video session and the time that the media player starts playing video frames. We also call *rebuffering delay* to be the video player interruption (blocking) time before it resumes.
- **Buffering ratio:** When the player buffer is empty –there is no more video frames to display at the screen –, the video stops (interruption or blocking). This event is called a *buffer starvation* or *buffering*. The buffering ratio is the ratio of the time the video player spends in blocking to the play time (including the blocking time).
- **Rate of buffering:** it is the frequency at which buffering events occur during the video session. it is computed as the ratio between the number of video player interruptions and the duration of the session.
- **Average bitrate:** The video quality is denoted by the bitrate in kilobits per second (kbps). In Variable Bit Rate (VBR) video or multiple bitrate video, the average bitrate is the mean quality of the streamed video. It is the time average of the bitrates played during the session weighted by the time duration each bitrate was played.
- **Number of bitrate switching:** it denotes the number of times the video quality changes during the session.

As mentioned before, the context and human also influence the Quality of experience. Thus, there exist external factors influencing the user perceived quality. Further, QoE metrics are interdependent. Therefore mapping the metrics to the user perceived quality which reflects the user satisfaction is a challenging task.

2.2.3 QoE measurement

The Quality of Service can be used to measure the QoE. It consists on measuring QoS metrics (bandwidth, delay, jitter, packet loss) and perform a mapping with QoE metrics [20, 21]. The mapping method usually uses machine learning approaches. Measuring the QoS parameters is simple and time saving because it is a network level measurement and it does not need the participation of the end users. However defining a proper relation between QoE and QoS parameters is difficult.

A traditional method of measuring the Quality of Experience is the Mean Opinion Score (MOS) [22]. The user gives a score from 1 (bad) to 5(excellent) which quantify its overall satisfaction of the system. The scores may not directly related to the system

parameters.

The previous metrics are replaced by new measurable engagement metrics such as viewing time, number of visits or customer return probability [23, 24]. These engagement metrics can be measured at the end user terminals without the participation of the clients by reporting the terminals logs or statistics. This is interesting for video streaming systems since we can easily predict and control the Quality of Experience by finding a relationship between the engagement and QoE metrics as follow.

$$Engagement \quad = \quad f(QualityMetric_i) \tag{2.1}$$

This function has been massively investigated by empirical studies [23, 24, 25, 26], in order to model the metric interdependencies and their relationship to the users engagement. They also identify the external factors that impact the user engagement: nature of the content (live video, VoD), type of device (PCs, mobile devices, TVs), connectivity (cable/DSL, wireless), user interest. It is shown that there is a non-monotonic relationship between the video bitrate and the user engagement [23]. Indeed, higher bitrate could lead to higher buffering ratio and higher initial delay.

2.2.4 QoE analytical computation

We once identify the relationship between the metrics and the engagement, perform analytical studies by computing QoE metrics in order to evaluate the system performance through QoE utility function. QoE metrics computation has been considered in the literature. Basically, it is based on the transient analysis of the video playout buffer which is modeled as a queue [27]. The arrival process is governed by the network and the departure process is function of the video playout rate.

A generalized method of computing QoE metrics is the use of buffer dynamics in order to derive ordinary differential equations (ODE) and partial differential equations (PDE). It uses the method of Ruin Theory [28] which studies insurer's vulnerability to insolvency. In [29], the network channel is modeled using Markov modulated arrival processes. The buffer is modeled as an M/D/1 queue and upper and lower bounds are provided on the minimum initial buffering (startup delay) required so that the playback interruption probability is below a desired level. Upper and lower bounds of the video playback interruption are also provided in [30, 31]. The distribution of the startup delay and the playout buffer starvation are computed using ordinary and partial differential equations in [32, 33]. In [32], the boundary condition of the startup delay is not a continuous function. Therefore, it is not possible to solve the PDEs. In [33], to avoid the boundary condition issue, the expected starvation probability before time t, given the initial channel state j and the initial queue length x,

$$W_j(x,t) = E[\mathbf{1}_{\xi(x)\leq t}|J(0) = j, X(0) = x] \tag{2.2}$$

is replaced by

$$W_j(x) = E[\mathbf{1}_{\xi(x)\leq T_\omega}|J(0) = j, X(0) = x] \tag{2.3}$$

where J is the set of channel state, $X(t)$ is the queue length at time t, $\xi = inf\{t \geq 0|X(t) < 0\}$ is the time of observing an empty buffer for the first time. The random variable T_ω represents the watch time that follows an exponential distribution. Indeed, letting the file size follows an exponential distribution, Eq. 2.3 gives an ordinary differential equation instead of PDE.

The playback buffer is modeled as a G/G/1 queue in [39]. The network is caracterize by the mean and the variance of the traffic arrival rate and the video playback rate. Then, diffusion approximation is used to derive closed-form expressions for the startup delay, the number of playback interruptions and the packet loss rate. The diffusion approximation method consists in replacing the discrete buffer size by a continuous process. Thus, the continuous process is modeled as a Brownian motion.

A recent method of QoE metrics computation is Ballot theorem [40]. It consists on counting process (votes) of two candidates during a ballot. The key feature of Ballot theorem is its simple expression to compute the probability that the counting process of the first candidate (e.g. arrival process) is strictly ahead of that of the second candidate (e.g. departure process). Ballot theorem has been used before for transient analysis of queueing systems [41, 42] and also for packet loss probability in M/M/1/K queue [43]. In [44], a discrete version of the theorem (Takacs Ballot theorem) is used to compute the probability of the playback starvation, the average playback interruption, the startup delay and the rebuffering delay. The arrival process is modeled by a Brownian motion considering a shared fast fading channel (Rayleigh channel) and the use of Proportional Fair (PF) and Round Robin (RR) schedulers. The video playback rate is deterministic, therefore the buffer is described by a M/D/1 queue. However, in [45], both continuous version using M/M/1 queue and discrete version of Ballot theorem are used to compute the metrics of the quality of experience. Similarly, in Chapter 4, we use Ballot theorem to analyse the performance of the Backward-Shifted Coding, the novel coding scheme that we introduce in this thesis. However, the Ballot theorem cannot be apply directly as seen in literature works.

2.3 HTTP Adaptive Video Streaming

In this section, we describe the Dynamic Adaptive Streaming over HTTP standard and review the quality adaptation methods proposed in the literature.

2.3.1 Dynamic Adaptive Streaming over HTTP

The first HTTP adaptive streaming solution is proposed by Move Networks in 2006 [46, 47]. Thereafter, it gains a particular attention in the industry level through commercial solutions. Among them, Microsoft Silverlight Smooth Streaming (MSS) [48] by Microsoft Corporation in 2008, HTTP Live Streaming (HLS) [49] by Apple Inc. in 2009 and Adobe HTTP Dynamic Streaming (HDS) [50] by Adobe Systems Inc. in 2010. The main issue with proprietary or commercial solutions is that they are jointly incompatible.

HTTP Adaptive Streaming (HAS) is first standardized in 2009 by the third generation partnership project (3GPP) Release 9 [51] in order to be used with the new UMTS LTE communication networks. The standard was improved by 3GPP in collaboration with the MPEG group [7]. Finally, the Dynamic Adaptive Streaming over HTTP standard was issued by MPEG in 2012 [8]. In recent years, MPEG-DASH has been integrated into new standardization efforts such as HTML5 Media Source Extensions (MSE) enabling the DASH playback via the HTML5 video and audio tag and HTML5 Encrypted Media Extensions (EME) enabling DRM-protected playback in web browsers.

Other proprietary solutions exist nowadays from Content Delivery Networks (CDN) such as Netflix, Akamai, Movestreaming, Amazon, etc. In addition, DASH Industry

Forum [52] has been formed in order to enable smooth implementation of DASH, one of their achievement is DASH-AVC/H264 recommendations [53] a recommendation of profiles and settings serving as guidelines for implementing DASH with H.264/AVC video.

The adoption of adaptive streaming based HTTP solution is motivated by the fact that non-adaptive streaming suffers important startup delays and buffering during the video session. This is due to the variation of the network capacity over the time. In addition, the use of HTTP allows efficient bypassing of NATs and Firewalls, which is not the case of Real-time Transport Protocol (RTP) over User Datagram Protocol (UDP) streaming solutions. Since in HAS, we use HTTP over TCP, a buffer is deployed at the client side to smooth out short-term transmission rate variation caused by TCP protocol. Other key targets of DASH are: simple advertisement insertion and the use of existing and cost-effective HTTP-based CDNs, proxies and caches.

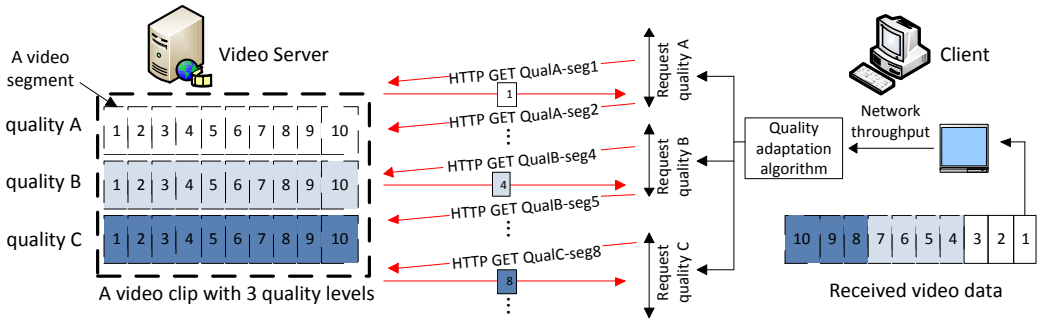

FIGURE 2.3: A typical DASH system [54]

In DASH, the media file is split into small segments of fixed duration, e.g., 2 seconds, which can be encoded at different bitrates or qualities (called representations). For instance, when using a single (multi) layer codec such as AVC (SVC), each segment has different versions (layers). The segments are provided on a web server and can be downloaded through HTTP standard compliant GET requests (Fig. 2.3). The adaptation to the bitrate is done on the client side for each segment, e.g., the client can switch to a higher bitrate - if bandwidth permits - on a per segment basis (Fig. 2.4). The temporal and structural relationships between segments are described in the Media Presentation Description (MPD) file. The MPD structure is detailed in Section 3.3.1.

DASH specifications include the definition of the media presentation description and segment formats, the conformance and reference software, the implementation guidelines, the segment encyption and authentication. DASH does not dictate the adaptation logic, this complexity is moved to the client side. The advantage is that client based streaming logic enables high scalability and flexibility. Therefore, the bitrate adaptation algorithms are not standardized in DASH, the aim is to choose a bitrate ensuring good video quality and prevent unnecessary video playback interruptions. We review the adaptations algorithms proposed in the literature in the next section.

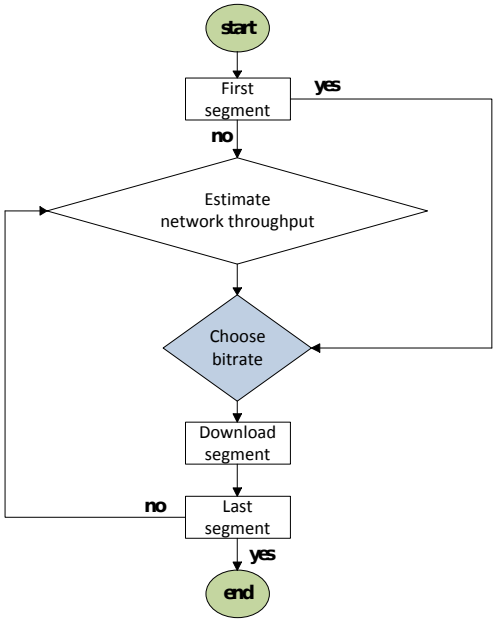

FIGURE 2.4: Flow diagram for adaptation algorithms for DASH

2.3.2 Overview of quality adaptation methods

The quality adaptation methods usually fall into two categories: the throughput-based algorithms (TBA) and the buffer-based algorithms (BBA). But in practice, they are hybrid since they both leverage on the estimated network throughput and the video player buffer occupancy level. They are independent of the video codecs, i.e., the following quality adaptations methods are valid for AVC, SVC or any equivalent international standard codec used to encode the different representations of the video.

Throughput-Based Algorithms

There exists different ways of estimating the available network throughput. We call *instant throughput* when the estimation is done on the last downloaded segment. This method is simple but the obtained throughput is highly fluctuating. When the estimation is done taking into account all the previous downloaded segments, we call it *smooth throughput*. This solution copes with the short-term fluctuations but it can cause late reaction of the client to a large throughput decrease. Other methods can be used such as the harmonic mean.

A pure throughput-based method is proposed in [54]. They integrate the available bandwidth measurement module in a proxy in order to detect the highest quality level that the current network conditions can support. Using subjective measurements, they showed that users prefer a gradual quality change between the best and the worst quality levels instead of an abrupt quality switching. Based on these findings, they proposed an adaptation algorithm that insert intermediate quality levels than directly switching to the target quality level provided by the proxy. An experimental evaluation on two commercial solutions (Smooth Streaming, Netflix) and one open source player (Adobe OSMF) is provided in [55]. They study how these players react to different network available bandwidth variations. Smooth Streaming player uses Microsoft silverlight version 4.0.50524.0. It starts with the lowest available bitrate and reaches

the highest possible bitrate in a few transitions. They showed that the player has two phases: the buffering phase where the segments downloading is not delayed and the steady state phase which delays segments downloading in order to maintain 30 seconds of video content in the buffer. The player reacts to persistent bandwidth variations with some delays and prefers to maintain a safety margin between the available bandwidth and the requested bitrate. Thus, it does not risk the playback interruption. For short-term variations, the player reacts only when the throughput decreases by decreasing the bitrate. Netflix player has a long startup delay (13 seconds). It starts from the lowest bitrate to reach the highest possible one by requesting a number of segments from all the available bitrates. It reaches the steady state with about 300 seconds of video content. The player is aggressive since it prefers to use bitrate higher than the available throughput. Under short-term variations the player does not decrease the bitrate when throughput decreases. On the other hand, it increases the bitrate when the throughput increases, thus causing more quality variations. The open source Adobe OSMF player version 10.1.102.64 was not stable because it oscillates between the lowest and the highest bitrates. This is essentially due to the fact that it starts with the lowest bitrate and jumps at the highest possible one in only one transition. Further, the target player buffer level is less than 10 seconds. Understanding the pros and cons of these proprietary solutions can be very useful to design algorithms in DASH.

Other throughput-based algorithms include the buffer occupancy in their bitrate selection decision. Fuzzy Logic Controller is used in [56] for the adaptation algorithm. The input parameters (normalized mismatch between the bitrate and the throughput, playback buffer occupancy) and the output parameters (the decision of changing the bitrate, the need of downloading segments with delay) are converted to a set of linguistic variables. For example the buffer occupancy can be converted to four levels: low, medium, high, full. In the same way, the decision of changing the bitrate can be converted to the following variables: increase, no-change, decrease, large-decrease. In [57], the client-side buffered video time is used as feedback signal and the bitrate is smoothly increased as the available network bandwidth increases. It is promptly reduced in response to bandwidth decrease. Adaptive bitrate streaming is addressed as a model-based predictive control problem in [58, 59]. The mathematical formulation is based on the quality of experience including the following metrics: average video quality, average quality variations, total rebuffer time. The goal is to look for the class of algorithms that maximize the quality of experience. They showed that the buffer-based methods outperform the throughput-based ones. They also showed that by combining the buffer occupancy with bandwidth predictions, their proposed model predictive control algorithm is closed to the optimal solution. Several quality adaptation algorithms are compared in [60] in the context of live streaming. They introduce another type of perceptual quality which is Just Noticeable Difference (JND) in order to know how the changes of bitrate impact end users perception. They evaluate how the preparation of representations sets affect the behavior of the adaptation methods. They showed that each evaluated method has its pros and cons, and has to be used in a specific context depending on the users requirements. For example, thresholded buffer method is suitable for on-demand streaming. If a small number of switches and a small deviation of bitrates are the most important, the conservative method should be selected, i.e., gradual change on the bitrate.

Buffer-Based Algorithms

Adaptation based on buffer aims at keeping the buffer occupancy at a desired level. It is usually a thesholded buffer method since buffer thresholds are set and the bitrate changes according to the level of the buffer. Three buffer thresholds are considered in [61]: B_1, B_2, and B_3. The goal of the algorithm is to keep the buffer size between B_2 and B_3 by controlling the bitrate. In order to minimize the startup delay, the lowest representation is selected for the first segment. By doing so, the video will remain at the lowest quality for a long time. So they introduce a startup phase at the beginning of the video session. It consists of selecting the next higher representation as long as the bitrate is below a certain percentage of the estimated throughput. They showed that the proposed algorithm performs well under challenging network conditions. It also exhibits a stable and fair behavior when multiple clients share the same connection. A similar thresholded buffer method is proposed in [62, 63]. Since the segments may have different sizes, they add the size of each segment in the MPD file and account for this in the throughput estimation. The throughput estimation is performed with weighted harmonic mean. They use the weighted harmonic mean download rate of the previous segments to predict the time to download the next segment. They showed that accounting for segments sizes improves the video quality. In [64, 65], two buffer thresholds are considered meaning the reservoir r and the cushion c. An adjustment function is used to select the bitrate when the buffer occupancy is between r and c. As in [61] the algorithm consists on two phases: the startup and the steady state phase. This algorithm can be considered as a pure buffer-based algorithm since the capacity estimation is unnecessary in the steady state. They showed that compared to Netflix default algorithm, it reduces the buffer rate by 10-20% while delivering a similar average video quality and a higher video quality in steady state. Due to its performances, our proposed buffer-based adaptation algorithm in Chapter 5 will be based on this algorithm.

The main issue of using throughput estimation in video quality adaptation is the possible throughput prediction errors. It is showed in [66] that existing streaming algorithms achieve between 69-86% of optimal quality if we assume to know the bandwidth for the entire video session. Since it is not practical to know such information they assume knowing the available bandwidth for a few seconds in the future. They showed that bandwidth predictions combined with rate smoothing functions allow to improve users QoE. A new algorithm is introduced in [67] which adapts video quality based on predictions of the wireless channel state. They anticipated poor network channel conditions like in non-coverage area by reducing the video quality in advance. In the same way, channel state information in mobile networks is considered in [68] to design adaptation algorithms. They formulate buffer aware policy problem using Markov Decision Process (MDP). They showed that knowing the channel state in advance will increase the benefits of HAS policies.

The above proposed algorithms sometimes fail to satisfy several users sharing the same link because of synchronization problems due to underestimation or overestimation of the available bandwidth. Therefore the bitrate can highly oscillate. Some approaches work around this by designing better scheduling strategies. AVIS and FESTIVE (Fair, Efficient and Stable adapTIVE) frameworks are proposed in [69] and [70] respectively. They are a resource management frameworks that schedule HTTP adaptive video flows in cellular networks. They are effective in allocating the resources of a base station

across multiple adaptive video flows and effectively balance between the three important goals that are: fairness between the users, stability of the video quality for each user and efficiency of the network resource utilization. NOVA [71] also performs joint optimization of network resource allocation and video quality adaptation. It maximizes users QoE by realizing tradeoffs among mean video quality, temporal variability in quality and fairness incorporating user preferences on rebuffering and cost of video delivery. In [72], scheduling and prediction are combined. They address resource allocation for the wireless downlink of a cellular network when future knowledge about the achievable throughput is available. Using linear programming they study the problem of optimal resource allocation in order to maximize average video quality.

In the above mentioned works, the adaptation is systematically performed in a single segment. In Chapter 5, we introduce new bitrate adaptation algorithms that select the bitrates of two superposed segments simultaneously.

2.4 Video objects caching

In this section we review existing caching policies and show that these caching policies fail to give plain satisfaction in the context of adaptive video streaming.

2.4.1 Overview of caching policies

Caching techniques are used in various applications such as computer networks, databases, operating systems, web pages, etc. Caches are small size storage devices (memory cache, Internet browser cache, cache server) which hold copies of contents stored in a main storage device. In computer networking, by caching contents close to the users, we reduce network traffic, latency and offload the servers. Indeed, when users request a content which is already in the cache, there is no need to retrieve it from the distant server, thus saving time and bandwidth. Since the sizes of caches are limited, all the contents cannot be stored. The caches rely on a *caching policy* to decide whether a content should be stored and what content should be discarded. A caching policy is determined by its replacement policies: decision to store or not the content (insertion) and which content is evicted from the cache (eviction). There exists, in the literature, a vast number of different policies to manage caches.

LRU: Least Recently Used [73]. LRU policy stores a new content at the roof of the contents in the cache until the cache becomes full. When the cache is full, LRU needs to make space before storing a new content. Thus, it evicts the content from the root of the cache and stores the new one at the roof. If a requested content is already in the cache, LRU just moves it at the roof. LRU insertion and eviction policies are summarized in Fig. 2.5.

LFU: Least Frequently Used [74]. LFU policy accounts for the content request frequency, i.e., the total number of requests for that content. It stores a content in the cache if and only if the frequency of that content is higher than the lowest frequency in the cache. The content with the lowest frequency is evicted from the cache. If a content already exists in the cache, its frequency is simply incremented by one unit. The contents with the higher frequencies are called popular contents. Thereby, LFU policy only stores popular contents in the cache respective to the cache size.

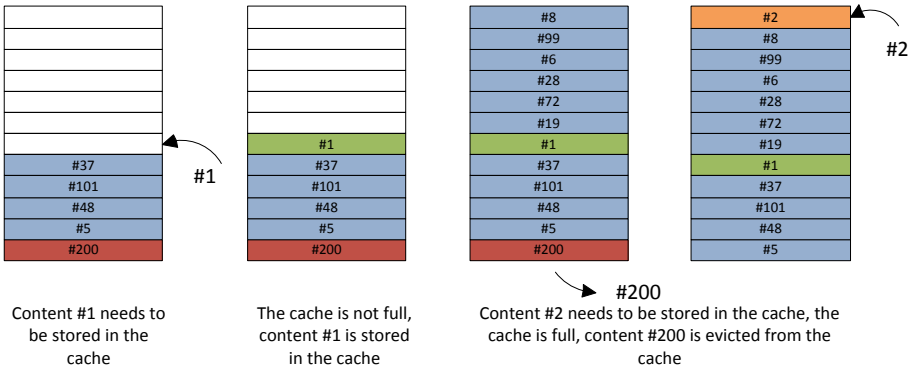

FIGURE 2.5: A simplified description of LRU insertion and eviction policy

FIFO: First In First Out [73]. FIFO policy stores a new content in the cache if it is not already in the cache. It evicts the oldest content from the cache to make space for a new one. The difference between FIFO and LRU policies is that FIFO policy do not update the position of a requested content which is already stored in the cache. On the other hand, if a requested content is already in the cache, LRU policy updates its position by moving it at the roof of the cache.

Random: The random policy has the same insertion policy as LRU or FIFO. It differs from those policies for the eviction policy. Indeed, to make space for a new content, it evicts randomly a content from the cache.

LRFU: Least Recently/Frequently Used [75]. LRFU policy is a spectrum of cache replacement policies between LRU and LFU. It combines benefits of LRU and LFU policies by specifying a weight for each requested content that decays exponentially over time. Each content is associated with a Combined Recency and Frequency (CRF) value. The computation of CRF uses a function $f(x) = \exp(-\gamma x)$ to weight the importance of the most recent occurrences for a content. The weighting parameter γ takes its value from 0 to 1. The computation of the CRF of a content b at time t_b is expressed as:

$$C_{t_b} = \begin{cases} f(0) + f(t_b - t_b').C_{t_b'} & if \quad b\ is\ known \\ f(0) & if \quad b\ is\ unknown \end{cases}$$

LRFU policy replaces the content with the minimum CRF value. If the content is already in the cache LRFU policy recalculates its CRF value and updates the time of the last request. If the content is not in the cache, it is retrieve from the distant server and its CRF value and the time of the last request are initialized. Then the content with the minimum CRF value (at the root of the cache) is replaced with the new content.

LRU-k: Recency and frequency is also combined in LRU-k policy [76]. It records the last k times each content was requested and evicts the one with least recent k^{th} access.

k-LRU: In k-LRU policy [77], contents are stored in the cache only after passing through $k-1$ virtual caches. There is a total of k caches, $k-1$ vitual caches and one physical cache. When a content is requested, if it is in cache i, it is moved to cache

$i+1$. If the content is not in any cache, it is stored to the first virtual cache. When the content reaches the k^{th} cache, it is stored physically in the cache. Every cache follows the LRU policy.

Belady's optimal policy: Belady's algorithm [78] stores requested contents in the cache until the cache becomes full. In order to make space for a new content, it evicts the content that has the highest next request time. Therefore, it needs future information about the request times of the contents. This algorithm is not implementable in a real system but it achieves theoretically optimal performance.

There exists caching policies that consider contents time varying popularities. In [79], the proposed caching policy uses a sliding window to discard the old requests. Thus it stores the top-k popular contents. The algorithm is compared with LRU, 2-LRU and Belady caching policies. The results show that the proposed algorithm outperform LRU and 2-LRU policies. Analytical caching performance is studied in [80, 77]. [80] derives hit rate approximatons for LRU cache with pre-filter under stationary and time varying popularities. In [77], by using existing results on LRU policy, they compute the cache hit probability for different caching policies.

2.4.2 Video caching in HTTP adaptive streaming

Caching is a suitable technique for video streaming systems [81]. Indeed, with the rapidly growing of video traffic over the Internet, it is essential to reduce the network latency and decrease the backhaul link traffic which is becoming the blottleneck in mobile networks [82]. In the context of 5G technologies, mobile edge caching solutions have recently been proposed in literature [83, 84, 85, 86, 87]. The existing caching techniques which were used for caching text and images can still be used in classical streaming where a video object is a unique file. But in the context of adaptive streaming a same video may have multiple representations (versions or layers) corresponding to different quality levels. Therefore existing caching techniques face the problem of representations caching. A basic method consists on considering each representation as a unique content and apply existing caching policies [88]. Obviously, this solution is not optimal. Should we cache all the representations of a given video or few of them? Which representations to cache? Some studies try to provide answers to these questions.

In [89], they investigate the problem of optimal content cache management for HTTP adaptive streaming over wireless networks. They consider a single video with several representations and a single cache. They look for the cache composition which achieves the optimal QoE where the QoE is defined as the ratio between the requested rate and the playback rate. They formulate the content cache management as a convex optimization problem and solve it using Lagrange multiplier method and alternative search algorithms to find the optimal number of cached files. DASH-aware scheduling algorithm for edge cache prefetching in Information Centric Networks (ICN) is proposed in [90]. By obtaining knowledge about the video segments using the Media Presentation Description, the algorithm prefetches the video segments in advance according to the current network capacity, the video rates and the client request history.

The video representations selection problem is studied in [91]. They consider video versions instead of layers. In order to compare their proposed algorithms, they use the mixed integer linear programming to obtain the optimal cache solution that maximizes the video quality. The mentioned caching strategies in [91] include *allQ*, *onlyLQ*, *onlyHQ*, *Qimpr* and *Partitioned*. However, the caches employ standard LRU replacement.

allQ algorithm caches all the representations for a requested video object. *onlyLQ* algorithm caches only the lowest quality for a requested video while *onlyHQ* algorithm caches only the highest quality. *Qimpr* algorithm caches the lowest quality for a requested video and increments the cached quality at each new request. *Partitioned* algorithm stores in the cache the same number of objects for each quality representation. The results showed that *Qimpr* and *Partitioned* algorithms are closed to the optimal solution given by the mixed integer linear programming problem because they store the most popular videos at high quality. The results are useful in the context of video versions (videos encoded using AVC for example) and LRU replacement policy. The case of video layers (videos encoded using SVC for example) is different since the layers are not independent such as versions.

Video versions and layers caching is studied in [92, 93]. In [92], they study the problem of optimizing the servicing cost of the network operators and the delivery delay for video requests of mobile users. They assume that the users specify the minimum video quality they are whishing to accept and the network provider goal is to minimize delay and cost while providing at least this quality. In fact, this requested quality is governed by the user channel state. They showed that when the user demand is homogeneous in terms of requested quality, the network operator can improve his balanced objective by using versions instead of layers. When the user demand becomes more diverse, layers can be more beneficial, as they allow more flexible caching. The mixed strategy, i.e., both versions and layers, performs the best. Pure versions, pure layers and mixed strategy are also compared in [93]. They showed that mixed strategy offers the best overall performance. They also showed that versions should be used if the requests for a specific video are for one quality level. If the video experiences multiple requests, the layers should be streamed and stored in the cache. These works focus on the comparison of pure versions, pure layers and mixed strategies rather than the optimization of the cache composition.

In Chapter 6 we propose caching policies to optimize the cache composition in the case of video layers and compare them with video versions.

2.5 Summary

We reviewed the background works in adaptive video streaming system. We saw that the notion of Quality of Experience (QoE) is more and more used to evaluate the performance of streaming systems instead of the Quality of Service (QoS). In fact, the quality of service is related to the system. Its metrics include network parameters such as bandwidth, delay, jitter or packet loss. Therefore, it does not related directly to the users engagement. Indeed, a good QoS does not necessarily guarantee that all customers experience the service to be good. In the other hand, the quality of experience represents the user-centric view of the performance of the system. It expresses directly the users perception of the system. This user perception can be quantified by metrics such as the startup delay, the buffering ratio, the rate of buffering, the average bitrate or the number of quality switching. To balance the tradeoff between those metrics, client-side quality adaptation algorithms are proposed in order to minimize the startup delay, buffering, quality switching and maximize the video quality.

Besides adaptive streaming, another proposed solution to increase performance of video delivery is mobile edge caching. Because of the recent growth of video traffic in the Internet, backhaul links are becoming the bottleneck of the transmission in mobile networks. Hence it is necessary to deploy caches in the network in order to offload web

servers and reduce latency. There is a lack of solutions to manage caches since our understanding of video caching is changed. In HTTP adaptive streaming, a same video is provided in multiple copies (representations), each of the copies corresponding to a different quality level. As a consequence, it is no longer sufficient to choose which video to cache, but also which of its representations to cache.

Chapter 3

Backward-Shifted Coding: a novel video transmission scheme

In this chapter, we explain the Backward-Shifted Coding (BSC) scheme. Since BSC is based on existing multi-layers video codecs and in particular the Scalable Video Coding, we first describe this video compression algorithm in details. Then, we give the basic idea of the Backward-Shifted Coding and how we can incorporate it in Dynamic Adaptive Streaming over HTTP (DASH), which is nowadays video transmission standard. We next discuss about the implementation of BSC protocol in DASH. Finally, we provide simulations to show that we hugely decrease the video playback interruption with this novel scheme.

3.1 Scalable Video Coding

Scalable Video Coding (SVC) [13] is one of the multiple extensions of H264/AVC. It is a multi-layer codec that was introduced in 2007. The particularity of this extension of AVC is its scalability. Although SVC has been an active research and standardization area for at least 20 years, it was not the first multi-layer codec. Indeed, scalability feature is found in prior video coding standards such as H262/MPEG-2 video, H263 or MPEG-4 visual. But they were rarely used because of drawbacks like the loss in the coding efficiency or the decoder complexity. Also their integration in traditional video transmission systems was a challenging task. Moreover, these standards have to compete with other techniques which also offer scalability such as transcoding and simulcast. Therefore the goal of SVC is to outperform these video coding standards and techniques in terms of coding efficienty and flexibility.

Fortunately SVC is an extension of AVC, thus it uses the key features used in AVC. This enables to achieve a significant improvement in compression efficiency and complexity compared to prior video coding standards. It integrates techniques like motion-compensated prediction, hierarchical prediction, intra prediction, inter-layer prediction, coarse-grain quality scalability, medium-grain quality scalability. Technical details on these techniques are provided in [94].

Using SVC, the video is encoded in one base layer (BL) and several enhancement layers (ELs). The base layer renders the video with the minimum quality and the quality is progressively increased by adding the enhancement layers. The enhancement layers offer different types of scalability: temporal (frame rate), spatial (image resolution), quality/SNR (image fidelity), Region of Interest and object-based scalability. Furthermore, the different types of scalability can be combined so that an enhancement layer supports two or more types of scalability. In the remainder, we assume that the enhancement layers support all the types of scalability.

Each enhancement layer is dependent on both the base layer as well as the enhancement layer below it to support the next higher level of video quality as shown in Fig.

3.1. There is a mapping between the quality levels and the layers. The base layer corresponds to the lowest quality level and the combination of all the layers corresponds to the highest quality. An important advantage of SVC is that it is natively resilient to

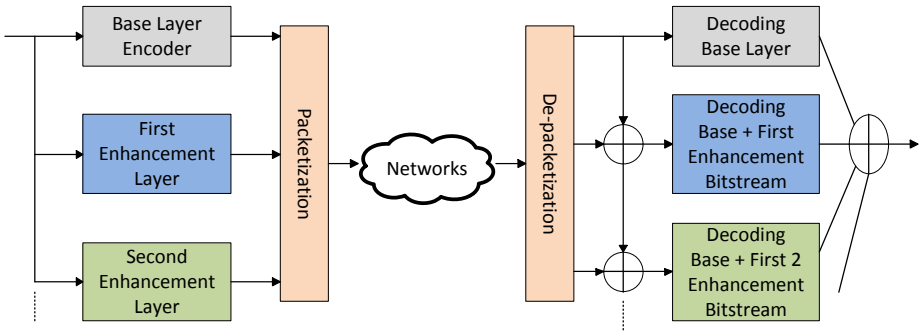

FIGURE 3.1: Coding and decoding layers with Scalable Video Coding

errors in the network, such as packet loss, because SVC codec transmits video frames using multiple video bitstreams rather than a single bitstream as done with AVC-based solutions. Then, SVC is able to compensate for missing data from frames in one video bitstream to another one. SVC also adds an encoding overhead compared to AVC. To understand this overhead we explain the packet structure of H264 standard. H264 standard has two conception layers: the Video Coding Layer (VCL) and the Network Abstraction Layer (NAL). The Video Coding Layer provides a coded signal from the video source signal while the Network Abstraction Layer formats these data to render NAL units. NAL units are packets, each containing a couple number of bytes. A packet contains a header (to specify the type of data) and a payload data representing the coded video. While the packet header is 1-byte in H264/AVC, SVC adds 3 additional bytes that are useful for the layers adaptations and the decoding process.

When used in HTTP adaptive video streaming, SVC offers the flexibility to the video client to download each video segment with the desired quality level. As shown in Fig. 3.2, the encoded video file exists in the video server with all the layers. The choice of the number of layers for each video depends on the content provider. Then, given the

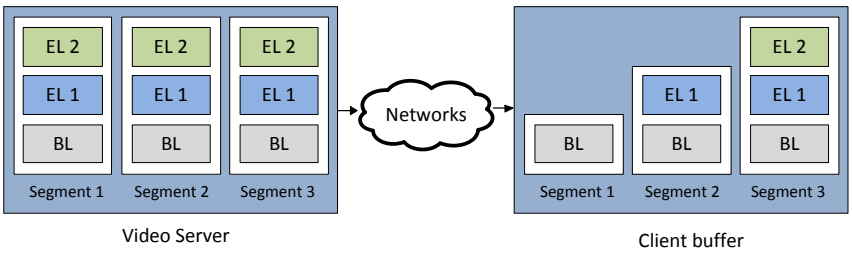

FIGURE 3.2: SVC in HTTP adaptive video streaming

network channel conditions, the video client decides either to download a segment with the base layer, or the base layer plus some enhancement layers. For instance, in Fig. 3.2, the client downloads the first segment with the lower quality, the second segment with the medium quality and the last segment with the highest quality.

The description of the Backward-Shifted Coding scheme will be based on Scalable Video Coding.

3.2 Backward-Shifted Coding (BSC)

In this section, we describe how the Backward-Shifted Coding (BSC) scheme can be used with any video codec that follows the H.264/SVC standard. Then we describe its integration into HTTP Adaptive Streaming (HAS) based on the Dynamic Adaptive Streaming over HTTP (DASH) standard [8]. We consider a video streaming system in which a video sequence is encoded using the scalable video coding.

In the rest of the thesis, we use the terms "playback interruption" and "buffer starvation" interchangeably.

3.2.1 Mapping from BSC scheme to video coding schemes

We assume that the video sequence has N frames numbered from 1 to N. Without loss of generality, we assume two layers in this section: the base layer and one enhancement layer. BSC is entirely client driven and compatible with all video coding standards supporting scalability. The main idea of BSC scheme is to shift the base layer (low quality) and the enhancement layer (high quality) of the same frame, so that, when an interruption of playback buffer occurs, the base layer frames can still be played. To each frame k, we add its base layer in some subsequent frame $k - \phi + 1$ as shown in Fig. 3.3 where ϕ is the offset between the two layers. Thus, if the buffer starvation happened at frame k, the playback retrieves the base layer frame from frame $k - \phi + 1$. In particular, we exploit temporal redundancy between subsequent frames in order to avoid the interruption of the playback buffer.

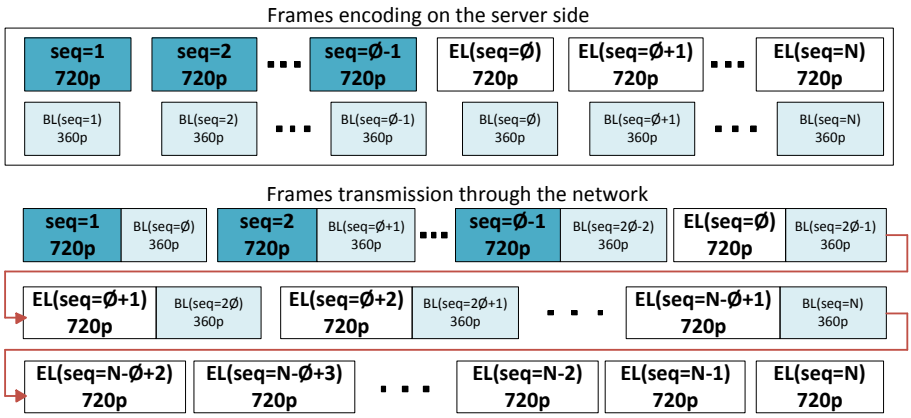

FIGURE 3.3: Using SVC in Backward-Shifted Coding

For example, let assume that the enhancement layer offers only the spatial resolution. Then, each frame has two resolutions: the base layer resolution (e.g., 640x352) and the enhancement layer resolution (e.g., 1280x704). The enhancement layer resolution is equivalent to the combined base and enhancement layers frame resolution. As mentioned in [13], the enhancement layer can be some regions of the original picture that will be used to build the borders of the base layer frame in order to increase its resolution. In BSC, each "frame" that span the network contains in reality the base layer and the

enhancement layer of two distinct frames. It is then natural to call it a block frame or simply a block.

As shown in Fig. 3.3, block k with $1 \leq k \leq \phi - 1$ contains: the complete frame k (base and enhancement layer of frame k) and the base layer of frame $k + \phi - 1$. On the other hand, block k with $k > \phi - 1$ contains: the enhancement layer of frame k and the base layer of frame $k + \phi - 1$.

At the user side, incoming NAL packets are reassembled into video frames by the decoder. Buffer starvation under BSC can happen at block k if the base layer block k is missing and the quality switching occurs when the enhancement layer is missing and the player finds only the base layer of block k. In the next chapter, we will consider this frames level video sequence and analyse the performance of BSC scheme.

3.2.2 BSC system description with DASH

We show that BSC scheme can be used at client side with DASH. Since with Dynamic Streaming over HTTP standard, the video quality is subject to change from segment to segment during the streaming session, we perform the integration of BSC in DASH at segments level rather than at frames level. We translate what we said previously on video segments because the idea of BSC remains the same. Therefore, we consider the video sequence as a set of consecutive segments numbered from 1 to K. The video sequence is encoded using the Scalable Video Coding to render multiple layers as shown in Fig. 3.4. Fig. 3.4 assumes 5 quality levels: 144p, 240p, 360p, 480p and 720p. The base layer

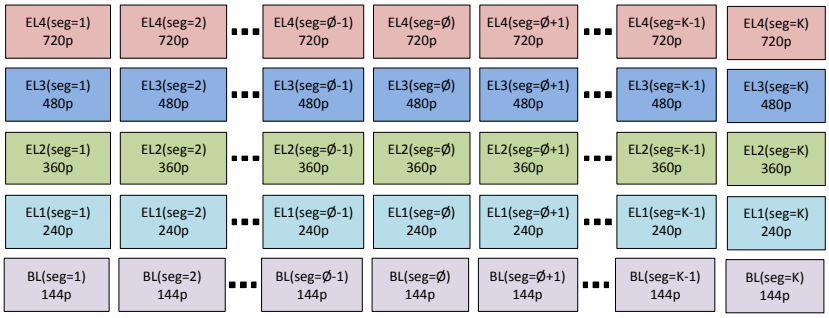

FIGURE 3.4: Video segments encoded with SVC

segments have a quality of 144p while all the layers together have a quality of 720p. As shown in Fig. 3.2 of Section 3.1, in DASH based SVC, each segment is requested with a given quality which corresponds to the base layer plus some enhancement layers. BSC acts in a different way. Indeed, with BSC, at time t_k, the segment k is requested with a given quality (base layer plus some enhancement layers). Then, at time $t_{k+\phi-1}$, some enhancement layers of segment k are requested with segment $k + \phi - 1$ in order to improve its quality as shown in Fig. 3.5. Hence, BSC sends a complete segment (base layer and possibly some enhancement layers) together with the enhancement layers of another segment. We call the first one, *low layer segment* and the second one is the *top layer segment*. The constant parameter ϕ is the offset between the *low layer segment* and the *top layer segment*. Thus, each segment k has its enhancement layers in segment $k + \phi - 1$. We call *block* k the combination of segment $k + \phi - 1$ (*low layer*) and enhancement layers of segment k (*top layer*). Note that the index of the *block* is also the index of the *top layer segment*.

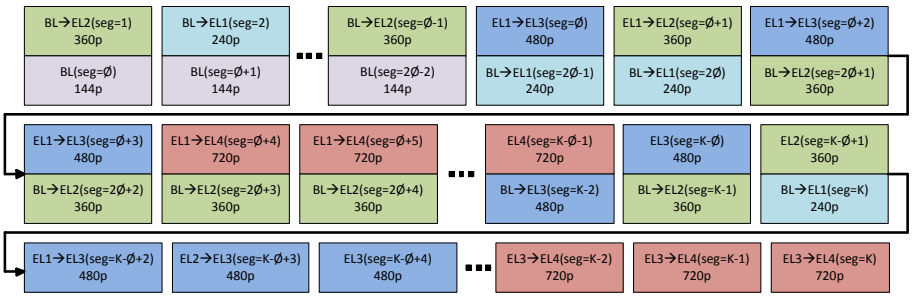

FIGURE 3.5: Segments transmission with Backward-Shifted Coding: the low layer segments contain the base layer (and possibly some enhanced layers) and are transmitted before the corresponding top layer segments, which follow after $\phi - 1$ blocks; the initial $\phi - 1$ blocks carry only low layer segments; the notation $BL \rightarrow EL_j$ indicates all segments BL, EL_1, EL_2, ..., EL_j and $EL_i \rightarrow EL_j$ indicates EL_i, EL_{i+1}, ..., EL_j

At the user side, incoming NAL packets are reassembled into video frames by the decoder. Since the two layers (low and top layers) of a given segment are sent separately, one can think that BSC performs repeated decoding process. It is not the case, because when the decoder receives the low layer segment through block k, it waits until block $k+\phi-1$ to get the top layer segment before performing the decoding process to render the segment with the quality of the top layer segment.

Since BSC has to manage two layers (low and top layers) together, it makes the system interesting but also challenging at the same time. Indeed, choosing the appropriate quality for those two layers is a difficult task because of the variability on network channel conditions. This will be addressed on Cchapter 5, where we consider BSC system with DASH standard and build quality adaptation algorithms.

3.3 BSC protocol implementation

In this section we discuss BSC protocol implementation. For this purpose, we first describe the Media Presentation Description file structure which is a key feature in DASH standard. Then, we introduce the modifications on DASH implementation that could take place with the Backward-Shifted Coding system.

3.3.1 Media Presentation Description

The Media Presentation Description (MPD) is an XML document that describes the media data (audio and video) available at the servers [95]. It contains information about media segments, the relationships between them and other metadata needed by the video clients to request the segments. The MPD is exchanged between the servers and the clients at the initialization of the video streaming session. The Media Presentation Description is a hierarchical data model as shown in Fig. 3.6.

Periods: The MPD file is organized in periods. Each period describes a part of the content with a start time and a duration. Then, the periods are used for the content splicing. They can also be used for advertisement insertion.

Adaptation sets: The period is organized in adaptation sets. An adaptation set contains a set of media streams. For example, it can contain one video adaptation

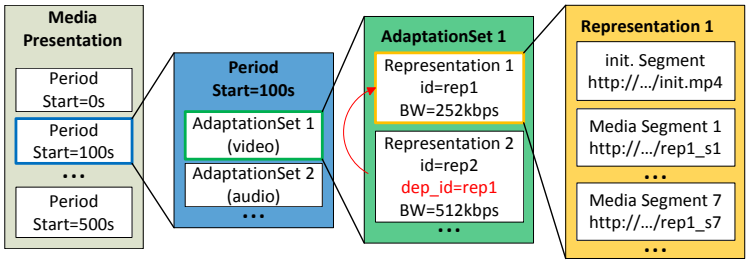

FIGURE 3.6: Media Presentation Description hierarchical structure

set and multiple audio adaptation sets (one for each supported language). It can also contain subtitles or arbitrary metadata.

Representations: An adaptation set contains the same content encoded in different ways. Each encoded content is called a representation. Representations are usually provided in multiple screen sizes and bandwidths. In the case of the scalable video coding, each layer is described by a different representation. The enhancement layers are considered dependent representations that depend on representations that contain lower layers, referred to as complementary representations. The dependency on other representation is indicated in the MPD file by the *dependency_id* (*dep_id* in Fig. 3.6), which indicates additional representations that are needed to be downloaded to be able to decode a dependent representation.

Segments: The representation contains the media segments that the video clients play. The media segments locations can be described using *BaseURL* for a single segment representation, a list of segments (*SegmentList*) or a template (*SegmentTemplate*). Information that applies to all segment can be found in a *SegmentBase*. The segments can be in separate files (common for live streaming) or they can be byte ranges within a single file (Video-on-Demand). The segments may also be subdivided in smaller subsegments which represent a set of smaller access units in the given segment. In this case, there is a **Segment index** available in the segment describing the presentation time range and byte position of the subsegments. Arbitrary quality switching between the representations is not possible at any point in the stream. DASH introduces the Stream Access Points (SAP) which are time instants where the switching can occur. For example, each segment typically begins with an IDR-frame (Instantaneous Decoding Refresh) that will allow switching between the representations (layers).

A simplified MPD XML file for SVC is represented in Listing 3.1 where each representation corresponds to a layer. The first representation is the base layer. The second representation corresponding to the first enhancement layer has the base layer as dependency layer. And the third representation has the base layer and the first enhancement layer as dependency layers.

```
<?xml version="1.0" encoding="UTF-8"?>
<MPD ...>
  ...
  <Period>
    <AdaptationSet ...>
      <Representation id="bl" ...>
        <BaseURL>video-500k.mp4</BaseURL>
      </Representation>
```

```
      <Representation id="el1" dependencyId="bl" ...>
        <BaseURL>video-1100k.mp4</BaseURL>
      </Representation>
      <Representation id="el2" dependencyId="bl el1">
        <BaseURL>video-1650k.mp4</BaseURL>
      </Representation>
    </AdaptationSet>
  </Period>
</MPD>
```

Listing 3.1: Excerpt of a simplified MDP for SVC

3.3.2 Modifications on DASH standard

As said before, Backward-Shifted Coding is entirely client driven, therefore it reuses all the components involved in the system architecture of DASH as shown in Fig. 3.7: servers (content providers), network (network operators) and clients (algorithms designers). It is worth to note that BSC system does not require additional cost from the content providers and the network operators. Thus it can be a competitive solution.

The servers are agnostic to delivered content since the complexity is moved to the client side in DASH architecture. The servers hold the description of the stored content (MPD) as well as the content. With the structure of the MPD file described in the previous section, we see that it is possible to request any segment layer any time. The stored files do not differ from DASH based SVC solutions. We can add the offset parameter at the top of the MPD document but it is not mandatory since the client can specify it in requesting the first BSC block (low and top layer segments).

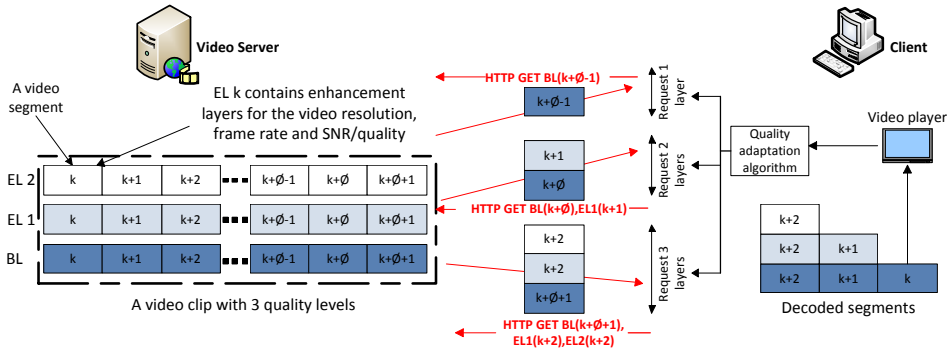

Figure 3.7: BSC system architecture based on DASH

As DASH, BSC uses HTTP over TCP since HTTP streaming has shown to be very valuable for streaming services. The increased latency of TCP protocol is acceptable, further, problems with firewall and NAT traversal are avoided. Therefore, the segments (low and top layer segments) are downloaded through HTTP standard compliant GET requests.

A slight modification has to be done at the client side in order to allow the fey feature of BSC: request two shifted layers simultaneously. In fact, the client should be capable to send two HTTP requests in parallel to get the low layer segment and the top layer segment within a single block. This is done by modifying the client side algorithm of DASH based SVC solutions.

3.4 Simulations

We perform simulations to have an idea of the video playback interruption with BSC and SVC systems. To that end, we consider BSC system at frames level without quality adaptation. We consider a tagged user downloading a video sequence of size N (in frames) encoded using the Scalabale Video Coding to render two layers: the base layer and one enhancement layer. We build an event driven simulator to simulate the frames arrival and departure processes into the video playback buffer. The empty buffer corresponds to the video playback interruption or the buffer starvation event.

We denote x to be the startup threshold, i.e., the number of frames we accumulate in the buffer before the player starts displaying video images. ρ is the traffic load, i.e., the ratio between the rate of the frames arrival and departure. We set x to 40 and the offset ϕ between the layers in BSC system is 50.

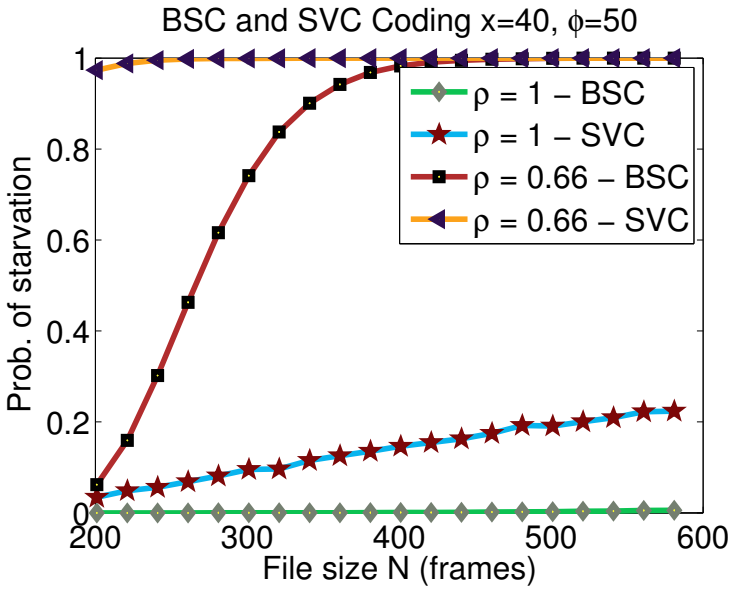

FIGURE 3.8: The probability of the playback buffer starvation for BSC and SVC coding schemes

Fig. 3.8 shows the probability of the playback buffer starvation for the two coding schemes: BSC and SVC. In both systems, the probability of the buffer starvation decreases when ρ increases. Indeed, ρ typically caracterizes the state of the network, as high values of ρ means good network state. For $\rho = 1$, the playback interruption is always avoided in BSC system while it reaches 20% for a file size of 600 frames in SVC system. The gap between the two curves increases as the traffic load decreases ($\rho = 0.66$). In the next chapter, we caracterize the probability of the playback buffer starvation analytically and confirm the benefits of BSC compared to SVC.

3.5 Summary

We describe the Backward-Shifted Coding scheme based on the Scalable Video Coding. BSC can use any video compression standard supporting the scalability feature, i.e., the video is encoded to render multiple layers: one base layer and more enhancement layers, those will increase the video quality once added to the base layer. The key

technique of BSC is to compound layers and defer the transmission of the enhancement layer segments (called top layer segments).

Since BSC is entirely client driven, it can be naturally adapted to Dynamic Adaptive Streaming over HTTP standard where the quality adaptation is performed at segments level. Therefore, the main challenge of BSC system is the quality selection of the two layers: low and top layer segments. Furthermore, BSC does not need modifications on DASH system architecture. Only the way requesting segments has to be rethink on the clients side.

We finally perform simulations considering video frames arrivals and departures processes, and we compute the probability of the video playback interruption. The results show that BSC hugely decreases the video playback interruption compared to an equivalent SVC system. That is the starting point of BSC performance analysis. In the next chapter, we will model the frames arrival and departure processes by standard probability distribution functions and analytically compute the metrics of Quality of Experience of BSC system.

Chapter 4

Evaluation of Backward-Shifted Coding Performance

We analyse the performance of the Backward-Shifted Coding scheme introduced in the previous chapter. Therefore, we adopt a QoE-based analysis since more and more content providers and network operators focus on quality of experience for the system operational efficiency [24]. The quality of experience is determined by the following metrics [25]: the join time of startup delay, the buffering ratio, the rate of buffering, the average bitrate and the quality variability. There is also the impact of some external factors which can only be determined by empirical studies [96]. In our analysis we compute the probability of the buffer starvation and the probability generating function of the starvation events by using the Ballot theorem [40]. Those two metrics are related to the buffering ratio and the rate of buffering. We next compute the startup delay and the average video quality using the quasi-stationary distribution of birth-death processes. We analyse the Backward-Shifted Coding without its integration in DASH, this will be the purpose of the next chapter. Finally, we propose a QoE objective function using the aforementioned metrics in order to evaluate BSC scheme. We pay a particular attention to the offset ϕ which is the most important parameter of the Backward-Shifted Coding. The entire analysis in this chapter is done at frames (or equivalently GoP) level.

4.1 Mathematical performance evaluation

In order to evaluate the efficiency of BSC using the SVC codec, we develop a novel performance evaluation for QoE based on the Ballot theorem [40]. The analysis of the buffer starvation is closely related to analyzing the busy period in transient queues but differs in two aspects: First, we aim to find the probability generating function of starvation events and not the queue size. Second, we do not assume a stationary arrival process.

4.1.1 Buffer starvation analysis

We call high quality frame the enhancement layer. The low quality frame corresponds to the base layer. We assume N to be the media file size in number of frames. When the streaming packets traverse the network, their arrivals to the media player are not deterministic due to the dynamic of the available bandwidth. The packets are reassembled by the decoder to render the video frames. We assume a Poisson distribution to describe the frames arrivals. After the streaming frames are received, they are first stored in the playout buffer. The interval between the service of two frames is assumed to be exponentially distributed so that we can model the receiver buffer as an M/M/1 queue. The exponential distributed assumption is not the most realistic way to describe

frame arrivals, but it reveals the essential features of the system, and it is the first step for more general arrival processes. In Section 4.3 we evaluate the performance of the Backward-Shifted Coding system by simulation, using different types of packet arrivals process such as the logistic process and the on-off process. The logistic process fits the video streaming traffic on the Long Term Evolution (LTE) networks according to [3].

The maximum buffer size is assumed to be large enough to exclude buffer overflows. ϕ is the offset between the high quality frame and its corresponding low quality frame (Fig. 4.1). A starvation happens when the playout buffer is empty.

We denote by λ the Poisson arrival rate of the frames, and by μ the Poisson service rate. We define $\rho = \lambda/\mu$ to be the traffic load. In a non-empty M/M/1 queue with everlasting arrivals, the rate at which either an arrival or a departure occurs is given by $\lambda + \mu$. The probability that this event is an arrival (departure) is p (q). where

$$p = \frac{\lambda}{\lambda+\mu} = \frac{\rho}{1+\rho}; \qquad q = \frac{\mu}{\lambda+\mu} = \frac{1}{1+\rho}$$

The buffer is initially empty. We let T_x be the initial buffering delay, in which x frames are accumulated in the buffer.

Probability of starvation

We present a frame level model to investigate the buffer starvation probability with the BSC. The analysis of the probability of starvation is built on the Ballot theorem.

Ballot theorem: In a ballot, candidate A scores N_A votes and candidate B scores N_B votes, where $N_A > N_B$. Assume that while counting, all the ordering (i.e. all sequences of A's and B's) are equally alike, the probability that throughout the counting, A is always ahead in the count of votes is $\frac{N_A - N_B}{N_A + N_B}$.

Since we set the value of ϕ at the beginning of the video session, the starvation can happen before the arrival of frame ϕ if $\phi > x$. So we have to investigate the probability of starvation by distinguishing the two cases: $\phi \leq x$ and $\phi > x$. Let $P_s^{<}(N, \phi, x)$ and $P_s^{>}(N, \phi, x)$ denote, respectively, the probability of starvation for $\phi \leq x$ and $\phi > x$. For

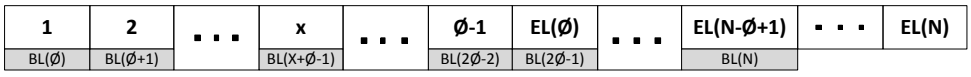

FIGURE 4.1: The BSC Coding for the offset $\phi > x$. The starvation can happen before the arrival of high quality frame ϕ since $x < \phi$.

$\phi \leq x$, the media player starts to work when the number of high quality frames in the buffer reaches x, which corresponds to $x + \phi - 1$ low quality frames stored in the buffer. Thus $P_s^{<}(N, \phi, x)$ is given by [45]:

$$P_s^{<}(N, \phi, x) = \sum_{k=x+\phi-1}^{N-1} \frac{x+\phi-1}{2k-x-\phi+1} \binom{2k-x-\phi+1}{k-x-\phi+1} \cdot p^{k-x-\phi+1} q^k \tag{4.1}$$

Let $P_{x,\phi}^k$ and P_x^k denote, respectively, the probability that the starvation happens exactly after the departure of frame k using BSC and without BSC. These probabilities are given

by:

$$P_{x,\phi}^{k} = \frac{x+\phi-1}{2k-x-\phi+1}\binom{2k-x-\phi+1}{k-x-\phi+1} \cdot p^{k-x-\phi+1} \cdot q^{k} \tag{4.2}$$

$$P_{x}^{k} = \frac{x}{2k-x}\binom{2k-x}{k-x} \cdot p^{k-x} \cdot q^{k} \tag{4.3}$$

Theorem 1. *For the offset $\phi > x$, the probability of starvation is given by:*

$$P_{s}^{>}(N,\phi,x) = P_{s1} + (1-P_{s1}) \cdot P_{s2} \tag{4.4}$$

where

$$P_{s1} = \sum_{k=x}^{\phi-2} \frac{x}{2k-x}\binom{2k-x}{k-x} \cdot p^{k-x} \cdot q^{k} \tag{4.5}$$

and

$$P_{s2} = \sum_{k=2\phi-2}^{N-1} \frac{x+\phi-1}{2k-x-\phi+1}\binom{2k-x-\phi+1}{k-x-\phi+1} \cdot p^{k-x-\phi+1} \cdot q^{k} \tag{4.6}$$

Proof. To build a link between the Ballot theorem and the video streaming system, we consider two events: A and B. The event A records the frames departures from the playback buffer while the event B records the frames arrivals into the playback buffer. Further, the frames are numbered from 1 to N and transmitted sequentially through the network in such a way that they arrive in order into the playback buffer. They are also played in that order.

For the case $\phi > x$, the starvation could happen before the arrival of frame ϕ. We define $E_{<\phi}$ and $E_{>\phi}$ to be the event that the starvation happens for the first time before ϕ and after ϕ, respectively. The event of starvation is $E_{<\phi} \cup (E_{>\phi} \cap \bar{E}_{<\phi})$; where $\bar{E}_{<\phi}$ is the complementary of $E_{<\phi}$ and is the event that no starvation happens before the arrival of frame ϕ. We have $P(E_{<\phi} \cup (E_{>\phi} \cap \bar{E}_{<\phi})) = P(E_{<\phi}) + P(E_{>\phi} \cap \bar{E}_{<\phi})$.

For the event $E_{<\phi}$, since we cannot use the low quality frames of the BSC coding because the starvation happens before ϕ, the probability of starvation is P_{s1}. We exclude in this sum $\phi - 1$ since the starvation cannot happen after the service of frame $\phi - 1$ because the frame ϕ (low quality) is already stored in the buffer.

Now we compute the probability of the second term $P(E_{>\phi} \cap \bar{E}_{<\phi})$. We have $P(E_{>\phi} \cap \bar{E}_{<\phi}) = P(E_{>\phi}/\bar{E}_{<\phi})P(\bar{E}_{<\phi})$. Since $\bar{E}_{<\phi}$ is the complementary of $E_{<\phi}$, $P(\bar{E}_{<\phi})$ is $1 - P_{s1}$. $E_{>\phi}/\bar{E}_{<\phi}$ is the event that a starvation happens for the first time after the arrival of frame $\phi-1$, given that the starvation does not happen before. Then, assuming that the starvation does not happen before $\phi - 1$, frame $x + \phi - 1$ will be played after $\phi - 1$ as shown in Fig. 4.2. At this instant, the low quality frame $2\phi - 2$ is already in the buffer, so the starvation cannot happen before the service of $2\phi - 2$. We use the Ballot theorem to compute the probability of the event $E_{>\phi}/\bar{E}_{<\phi}$ based on the low quality frames. The most important trick is the origin of the Ballot theorem, i.e., where to start the process of counting the frames arrivals and departures. The inappropriate method is to start the counting process just after the arrival of the frame $\phi-1$. At that moment, we do not know the number of departures that occur before. So we start the counting process when we have x high quality frames in the buffer, that correspond to the last low quality frame $x + \phi - 1$.

We define A_k to be an event that the buffer becomes empty for the first time when the service of frame k is finished (Fig. 4.2). All the events $A_k, k = 1, ...N$, are mutually exclusive. The event of starvation is the union $\cup_{k=2\phi-2}^{N-1} A_k$. We exclude in this union A_k for $k \in [1, 2\phi - 3]$ because we cannot have a starvation before the arrival of high quality frame $\phi - 1$. That corresponds to the low quality frame $2\phi - 2$. This union

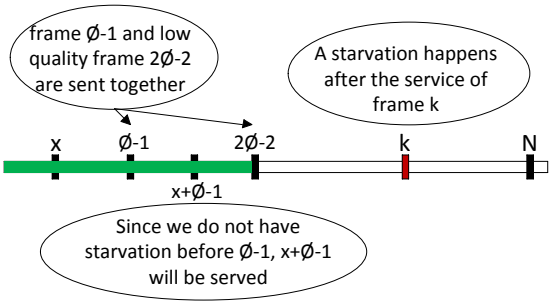

FIGURE 4.2: If the starvation does not happen before $\phi - 1$, then there will be no starvation untill the service of frame $2\phi - 2$.

of events excludes E_N because the empty buffer after the service of N frames is not a starvation. When the buffer is empty at the end of the service of the k^{th} frame, the number of arrivals is $k - x - \phi + 1$ after the startup delay process. The probability of having $k - x - \phi + 1$ arrivals and k departures is computed from the binomial distribution $\binom{2k-x-\phi+1}{k-x-\phi+1} \cdot p^{k-x-\phi+1} \cdot q^k$. For the necessary and sufficient condition of the event A_k, we apply the Ballot theorem. If we count the number of arrivals and departures when the playback starts, the number of departures is always greater than the number of arrivals. Otherwise, the empty buffer already happens before the k^{th} frame is served. According to the Ballot theorem, the probability of event A_k is computed by $\frac{x+\phi-1}{2k-x-\phi+1}\binom{2k-x-\phi+1}{k-x-\phi+1} \cdot p^{k-x-\phi+1} \cdot q^k$. Therefore, the probability of the event $E_{>\phi}/\bar{E}_{<\phi}$, is the probability of the union $\cup_{k=x+\phi-1}^{N-1} A_k$, given by Eq. (4.6). □

Probability generating function of the number of starvation events

In the BSC scheme, the starvation may happen for more than once during the file transfer. We are interested in the probability distribution of starvation, given the finite file size N. When $\phi \leq x$, we cannot have a starvation before the service of $\phi - 1$. In this case, the probability generating function is similar to that of [45] in replacing x by $x + \phi - 1$. The maximum number of starvations is $J = \lfloor \frac{N}{x+\phi-1} \rfloor$ where $\lfloor . \rfloor$ is the floor of a real number. We can see that the BSC scheme decreases considerably the number of starvations since the maximum number of starvations in M/M/1 system without BSC is $\lfloor \frac{N}{x} \rfloor$. It shows also the influence of the parameter ϕ on BSC system.

Now, we show how the probability generating function of starvation events can be derived using the Ballot theorem for the case $\phi > x$. We define a path as a complete sequence of frame arrivals and departures [45]. The probability of a path depends on the number of starvations. We consider a path with j starvations. To carry out the analysis, we start from the event that the first starvation takes place. We denote by k_l the l^{th} departure of a frame that sees an empty buffer. We notice that the path can be decomposed into the following mutually exclusive events:

- Event $\mathcal{F}(k_1)$: the buffer becoming empty for the first time in the entire path.
- Event $\mathcal{M}_l(k_l, k_{l+1})$: the empty buffer after the service of frame k_{l+1} given that the previous empty buffer happens at the departure of frame k_l.
- Event $\mathcal{L}_j(k_j)$: the last empty buffer observed after the departure of the last frame k_j.

We let $P_{\mathcal{F}(k_1)}$, $P_{\mathcal{M}_l(k_l,k_{l+1})}$ and $P_{\mathcal{L}_j(k_j)}$ be the probabilities of events $\mathcal{F}(k_1)$, $\mathcal{M}_l(k_l, k_{l+1})$ and $\mathcal{L}_j(k_j)$, respectively. We analyze the probabilities of these events step by step. We first compute the probability of having only one starvation. This probability concerns the two events: $\mathcal{F}(k_1)$ and $\mathcal{L}_1(k_1)$, i.e., the event that the buffer becomes empty for the first time after the service of the frame k_1 and the event that we do not observe an empty buffer after k_1 until the end of the video file (Fig. 4.3 and 4.4). The starvation

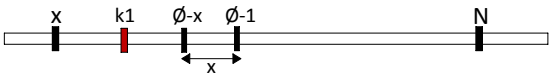

FIGURE 4.3: A starvation happens at $k_1 < \phi - x$.

can happen at k_1 before we use the BSC low quality frames, i.e., before the arrival of frame $\phi - 1$. In that case, the probability of starvation is given by $P_x^{k_1}$ of Eq. (4.2). If the starvation happens after the arrival of frame $\phi - 1$, then it necessarily happens

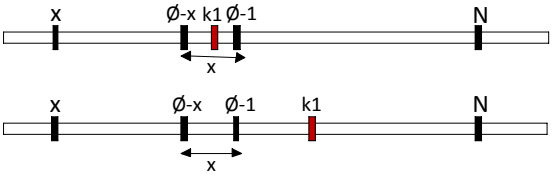

FIGURE 4.4: A starvation happens at $k_1 > \phi - x$.

after the service of frame $2\phi - 3$ since we cannot have a starvation between $\phi - 2$ and $2\phi - 3$. Then the probability of starvation is given by Theorem 1. So the probability distribution of event $\mathcal{F}(k_1)$ is expressed as

$$P_{\mathcal{F}(k_1)} := \begin{cases} 0, & if \quad k_1 < x \quad or \quad k_1 = N; \\ P_x^{k_1} & if \quad k_1 \in [x, ..., \phi - 2]; \\ 0, & if \quad k_1 \in [\phi - 1, ..., 2\phi - 3]; \\ (1 - P_{s1})P_{x,\phi}^{k_1}, & \\ if \quad k_1 \in [2\phi - 2, ..., N - 1]. & \end{cases} \tag{4.7}$$

where $P_x^{k_1}$, $P_{x,\phi}^{k_1}$ and P_{s1} are given by Eq. (4.2), Eq. (4.3) and Eq. (4.5) respectively. Given that the only starvation happens at k_1, what is the probability that no starvation happens until the end of the video file? That is the probability of the event $\mathcal{L}_1(k_1)$. We take the complement of starvation probability as the probability of no starvation for the file size $N - k_1$. We distinguish two cases (Fig. 4.3 and 4.4). The first case is that the starvation happens at k_1, and we still can have a starvation before the arrival of frame $\phi - 1$ (Fig. 4.3). Then, we use the probability of starvation of Theorem 1, $P_s^{>}(N - k_1, \phi, x)$ to compute the probability of having no starvation until the end of the video file. For the remaining case (Fig. 4.4), a starvation cannot happen before we use the low quality frames. Then, we use the probability of starvation, $P_s^{<}(N - k_1, \phi, x)$.

Finally, the probability distribution of event $\mathcal{L}_1(k_1)$ is expressed by

$$P_{\mathcal{L}_1(k_1)} = \begin{cases} 0, & if\ k_1 < x\ or\ k_1 = N; \\ 1 - P_s^{>}(N - k_1, \phi, x), & if\ x \leq k_1 < \phi - x; \\ 1 - P_s^{<}(N - k_1, \phi, x), & if \\ \phi - x \leq k_1 < \phi - 1\ or\ 2\phi - 2 \leq k_1 < N - x - \phi + 1; \\ 0, & if \quad \phi - 1 \leq k_1 < 2\phi - 3; \\ 1, & if \quad N - x - \phi + 1 \leq k_1 < N. \end{cases} \tag{4.8}$$

We denote by $P_s(j)$ the probability of having j starvations. For the case with one starvation, $P_s(1)$ is solved by

$$P_s(1) = \sum_{i=1}^{N} P_{\mathcal{F}(i)} P_{\mathcal{L}_1(i)} = \mathbf{P}_{\mathcal{F}} \cdot \mathbf{P}_{\mathcal{L}_1}^T \tag{4.9}$$

where ^{T} denotes the transpose. Here, $\mathbf{P}_{\mathcal{F}}$ is the row vector of $P_{\mathcal{F}(i)}$, and $\mathbf{P}_{\mathcal{L}_1}$ is the row vector of $P_{\mathcal{L}_1(i)}$, for $i = 1, 2, ..., N$. Now we compute the probability of having more than one starvation. A path with j starvations is composed of a succession of events

$$\mathcal{F}(k_1), \mathcal{M}_1(k_1, k_2), ..., \mathcal{M}_l(k_l, k_{l+1}), ..., \mathcal{M}_{j-1}(k_{j-1}, k_j), \mathcal{L}_j(k_j).$$

We have to solve the probability distribution of the events $\mathcal{L}_j(k_j)$ and $\mathcal{M}_l(k_l, k_{l+1})$. Suppose that there are j starvations after the service of frame k_j. Then, only the lower bound of k_j changes in the event $\mathcal{L}_j(k_j)$ from $\mathcal{L}_1(k_1)$. This lower bound corresponds to the extreme case where the j starvations take place consecutively. Let e be the number of starvations that we can have before the arrival of frame $\phi - 1$ in the extreme case. So $e = \lfloor \frac{\phi - 2}{x} \rfloor$. We distinguish two cases where $e \geq j$ and $e < j$ (Fig. 4.5). If $e \geq j$, then

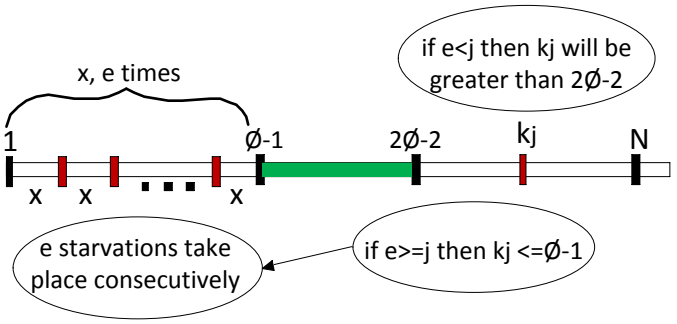

FIGURE 4.5: The lower bound of k_j in case we have j starvations.

all the j starvations will happen before the service of frame $\phi - 1$ and the lower bound of k_j is jx. Then, we find the expression of $\mathcal{L}_j(k_j)$ in replacing the lower bound x by jx in the expression of $\mathcal{L}_1(k_1)$. If $e < j$, then the remaining starvations will happen after the service of frame $2\phi - 2$ since we do not have starvation between $\phi - 1$ and $2\phi - 2$.

Hence, the probability distribution of event $\mathcal{L}_j(k_j)$ is

$$P_{\mathcal{L}_j(k_j)} = \begin{cases} 0, \\ if\ k_j < 2\phi - 2 + (j-e)(x+\phi-1)\ or\ k_j = N; \\ 1 - P_s^{<}(N-k_j, \phi, x), \\ if\ 2\phi - 2 + (j-e)(x+\phi-1) \leq k_j < N - x - \phi + 1; \\ 1, \quad if \quad N - x - \phi + 1 \leq k_j < N. \end{cases} \tag{4.10}$$

We now compute the probability of the event $\mathcal{M}_l(k_l, k_{l+1})$. After frame k_l is served, the l^{th} starvation is observed. We compare k_l to e to obtain the position of k_l. If $e < l$, then k_l should not be less than $2\phi-2+(l-e)(x+\phi-1)$ in order to have l starvations. Also, k_{l+1} must satisfy $k_l + (x+\phi-1) \leq k_{l+1} < N - (j-l-1)(x+\phi-1)$ because of the startup delay after k_l and the fact that we have j starvations in total. In that case, $P_{\mathcal{M}_l(k_l,k_{l+1})}$ is expressed as $\frac{x+\phi-1}{2k_{l+1}-2k_l-x-\phi+1}\binom{2k_{l+1}-2k_l-x-\phi+1}{k_{l+1}-k_l-x-\phi+1}p^{k_{l+1}-k_l-x-\phi+1}q^{k_{l+1}-k_l}$. If $e \geq l$, then we have $lx \leq k_l \leq \phi-1$. For $k_l+x \leq k_{l+1} < \phi-1$, $P_{\mathcal{M}_l(k_l,k_{l+1})}$ is expressed as $\frac{x}{2k_{l+1}-2k_l-x}\binom{2k_{l+1}-2k_l-x}{k_{l+1}-k_l-x}p^{k_{l+1}-k_l-x}q^{k_{l+1}-k_l}$. Since the $(l+1)^{th}$ starvation cannot happen between $\phi-1$ and $2\phi-2$, for $2\phi-2 \leq k_{l+1} < N-(j-l-1)(x+\phi-1)$, the probability of the event $\mathcal{M}_l(k_l, k_{l+1})$ is expressed as the probability for the case $e < l$. We denote by $\mathbf{P}_{\mathcal{M}_l}$ the matrix of $P_{\mathcal{M}_l(k_l,k_{l+1})}$ for $k_l, k_{l+1} \in [1, N]$. The probability of having $j(j \geq 2)$ starvations is given by

$$P_s(j) = \sum_{k_1=1}^{N}\sum_{k_2=1}^{N}, \ldots, \sum_{k_{j-1}=1}^{N}\sum_{k_j=1}^{N} P_{\mathcal{F}(k_1)} \cdot P_{\mathcal{M}_1(k_1,k_2)}, \ldots,$$

$$P_{\mathcal{M}_{j-1}(k_{j-1},k_j)} \cdot P_{\mathcal{L}_j(k_j)} = \mathbf{P}_{\mathcal{F}} \cdot \left(\prod_{l=1}^{j-1} \mathbf{P}_{\mathcal{M}_l}\right) \cdot \mathbf{P}_{\mathcal{L}_j}^T. \tag{4.11}$$

Then, we can write the probability generating function (p.g.f) $G(z)$ by

$$G(z) = E(z^j) = \sum_{j=0}^{J} P_s(j) \cdot z^j. \tag{4.12}$$

Asymptotic Property: When the file size is large enough, the probability of starvation can be approximated by the asymptotic behavior of the starvation probability for the case $\phi \leq x$, which is given by

$$\lim_{N\to\infty} P_s(\phi, x) := \begin{cases} 1 & \text{if } \rho < 1; \\ \exp\left(\frac{(x+\phi-1)(1-2p)}{2pq}\right) & \text{otherwise .} \end{cases}$$

Furthermore, the average time interval between two starvations for $\rho < 1$ is given by $\frac{x+\phi-1}{\lambda(1-\rho)}$.

4.1.2 Computing the average video bitrate

In this section, we compute the average video quality of the BSC system. For this purpose we model the system as a continuous-time Markov birth-death process with an absorbing state which corresponds to the starvation event (Fig. 4.6). Then, we compute the amount of time spent in each quality level. We call low bitrate, the bitrate of the base layer frames (or low quality frames) and high bitrate, the bitrate of the combined base and enhancement layers frames (or high quality frames).

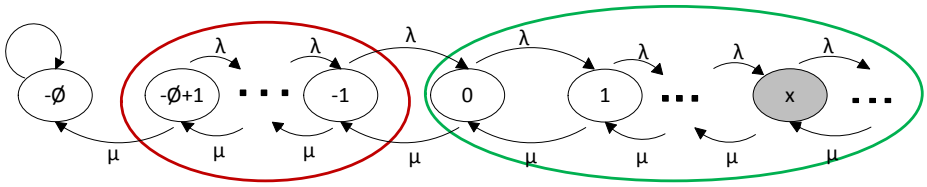

FIGURE 4.6: Markov model of the quality switching mechanism, denoting the difference between the number of high quality and low quality frames in the playback buffer with the absorbing state $-\phi$.

Let $\mathfrak{H}$ and $\mathfrak{P}$ be, respectively, the sequence number of the last high quality frame in the buffer and the sequence number of the last frame that was displayed at the screen. The state i of the Markov process is $\mathfrak{H} - \mathfrak{P}$. If $\mathfrak{P} \leq \mathfrak{H}$, the state i is non-negative and there is exactly $\mathfrak{H} - \mathfrak{P}$ available high quality frames in the buffer. Otherwise, if $\mathfrak{P} > \mathfrak{H}$, the state i is negative. In that case, there is no more available high quality frames and the number of available low quality frames is $\mathfrak{H} - \mathfrak{P} + \phi$. For instance if the process is in the state $-\phi + 1$ (i.e., $\mathfrak{H} - \mathfrak{P} = -\phi + 1$), it remains exactly 1 low quality frame in the buffer as shown in Fig. 4.6.

The infinitesimal generator Q of the process is a tridiagonal matrix where the first element of the diagonal and upon the diagonal is 0 (due to the absorbing state $-\phi$).

$$Q = \begin{pmatrix} 0 & 0 & 0 & \cdots & \cdots & \cdots & \cdots \\ \mu & -(\lambda+\mu) & \lambda & 0 & \cdots & \cdots & \cdots \\ 0 & \mu & -(\lambda+\mu) & \lambda & 0 & \cdots & \cdots \\ \vdots & 0 & \mu & -(\lambda+\mu) & \lambda & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

First, we compute the time to absorption starting from initial state x. To do this, we substitute transition to the absorbing state $-\phi$ by transition to the initial state x (Fig. 4.7), whenever the resulting Markov chain is ergodic and admits a stationary regime $\mathbf{q} = (q_{-\phi+1}, q_{-\phi+2}, ..)$. $\mathbf{q}$ is given by solving the balance equations of the resulting Markov chain

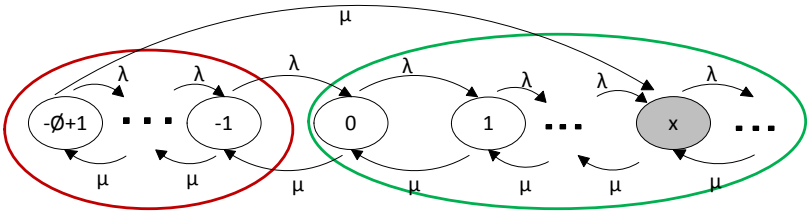

FIGURE 4.7: Markov model of the switching mechanism, denoting the difference between the number of high quality and low quality frames in the playout buffer.

$$\begin{cases} \mu q_{-\phi+2} & = & (\lambda+\mu)q_{-\phi+1} \\ \lambda q_{i-1} + \mu q_{i+1} & = & (\lambda+\mu)q_i, i \neq x+1 \\ \lambda q_{x-1} + \mu(q_{x+1} + q_{-\phi+1}) & = & (\lambda+\mu)q_x \\ \sum_{i=-\phi+1}^{\infty} q_i = 1 \end{cases} \tag{4.13}$$

$$\begin{aligned} q_i &= q_{-\phi+1}\frac{1-\rho^{i+\phi}}{1-\rho}, \quad i = -\phi+1, -\phi+2, .., x \\ q_i &= q_{-\phi+1}\frac{1-\rho^{x+\phi}}{1-\rho}\rho^{i-x}, \quad i = x+1, x+2, ... \end{aligned}$$

Thus, we have

$$\begin{cases} q_{-\phi+1} & = & \frac{1-\rho}{\phi+x} \\ q_i & = & \frac{(1-\rho^{i+\phi})}{(\phi+x)}, \quad i = -\phi+1, .., x \\ q_i & = & \frac{(1-\rho^{x+\phi})}{((\phi+x)}\rho^{i-x}, \quad i = x+1, x+2, ... \end{cases} \tag{4.14}$$

Let $E[S_i]$ be the expected time spent in state i before the process reaches the absorbing state. From [97], we have

$$E[S_i] = q_i E[\tau_x]$$

where $E_{[\tau_x]}$ is the time to absorption into state $-\phi$ starting from the initial state x. According to [98], the mean time to absorption into state $-\phi$ from the initial state x is given by

$$E[\tau_x] = \frac{x+\phi}{\mu-\lambda} \tag{4.15}$$

Thus the expected time spent in state i before the process reaches the absorbing state is given by

$$E[S_i] = q_i\frac{x+\phi}{\mu-\lambda} \tag{4.16}$$

where q_i is given by Eq. (4.14). Thus, the time spent in the low bitrate and in the high bitrate are given, respectively, by

$$T_{\mathcal{L}} = \sum_{i=-\phi+1}^{-1} E[S_i] \text{ and } T_{\mathcal{H}} = \sum_{i=0}^{\infty} E[S_i] \tag{4.17}$$

Let $b_{\mathcal{L}}$ and $b_{\mathcal{H}}$ be the low bitrate and the high bitrate, respectively. Hence, the average bitrate is

$$b_{avg} = \frac{T_{\mathcal{L}} b_{\mathcal{L}} + b_{\mathcal{H}} T_{\mathcal{H}}}{T_{\mathcal{L}} + T_{\mathcal{H}}} \tag{4.18}$$

Let us now compute the distribution of period of time during which the video playback quality is optimal, named B_H. This period corresponds to the duration of time that the process starting from state 1, stays continuously away from state 0. Using the analysis from the busy period in M/M/1, the expected and variance of B_H is given, respectively, by [98]

$$E[B_H] = \frac{1}{\mu-\lambda}, \quad \text{and } V(B_H) = \frac{(1+\rho)}{\mu^2(1-\rho)^3}$$

4.1.3 Startup delay and rebuffering delay

The expected initial buffering delay is $\frac{x}{\lambda}$ and the expected rebuffering delay is $\frac{x+\phi-1}{\lambda}$. Note that the BSC scheme increases the start-up delay compared to an equivalent non-BSC system because of the offset ϕ. Indeed, we send the enhancement layer ϕ after the complete frame $\phi - 1$. But in practice, this delay does not exceed a couple of seconds, then it does not have a huge impact on the overall quality of experience. However, ϕ impacts the rebuffering delay, i.e., the waiting time after a starvation event. The player starts when we accumulate x high quality frames in the buffer. So a large value of ϕ increases the amount of rebuffering time.

4.2 Evaluation of the QoE

We will show now how the BSC scheme improves the metrics of the quality of experience. The parameter ϕ impacts the startup delay and the starvation. We first confirm this with the analytical model and show how to choose a suitable value for ϕ. Then, we propose a QoE objective function using the aforementioned metrics to compare BSC system with a non BSC system, i.e., a simple SVC system.

4.2.1 The offset ϕ

To set the value of the offset ϕ, we should know the probability of starvation of a non BSC system according to the file size N. This probability is given in [45] by

$$P_{starvation} = \sum_{k=x}^{N-1} \frac{x}{2k-x} \binom{2k-x}{k-x} \cdot p^{k-x} \cdot q^{k} \tag{4.19}$$

where x is the startup delay threshold. It also corresponds to the probability of playing the low quality frames for the first time in BSC system since a starvation in a non BSC system is equivalent to a quality switching in BSC system. The surrounded region in Fig. 4.8 corresonds to the set of values of N where the risk of the playback interruption is negligible (ϕ can be set up to 80 frames in the Fig. without risk of starvation).

4.2.2 QoE objective function

We define an objective QoE cost function $C(x, N)$ for a given user, which includes the startup delay, the number of playback interruptions during the video session and the average video quality. These metrics are weighted by three coefficients γ_1, γ_2 and γ_3, which allow us to balance the tradeoff among the QoE metrics according to user preferences.

The weights γ_1 and γ_2 are preceded by a non-negative sign because the smaller the startup delay and the number of starvation are, the better the QoE is. Implicit tradeoff exists between the two metrics and the initial buffering time is preferred to the starvation by around 90% of users [99]. Although users have a very low tolerance for playback interruptions [100] they also only accept a start-up delay between 5 and 15 seconds, depending on the duration of the entire video [99].

The weight γ_3 is preceded by a negative sign because the higher the average quality, the better the QoE. According to this reasoning, we propose the cost function

$$C(x, N) = \gamma_1 \cdot E[T_x] + \gamma_2 . P_s - \gamma_3 \cdot \sum_{i=1}^{r} w_i T_i \tag{4.20}$$

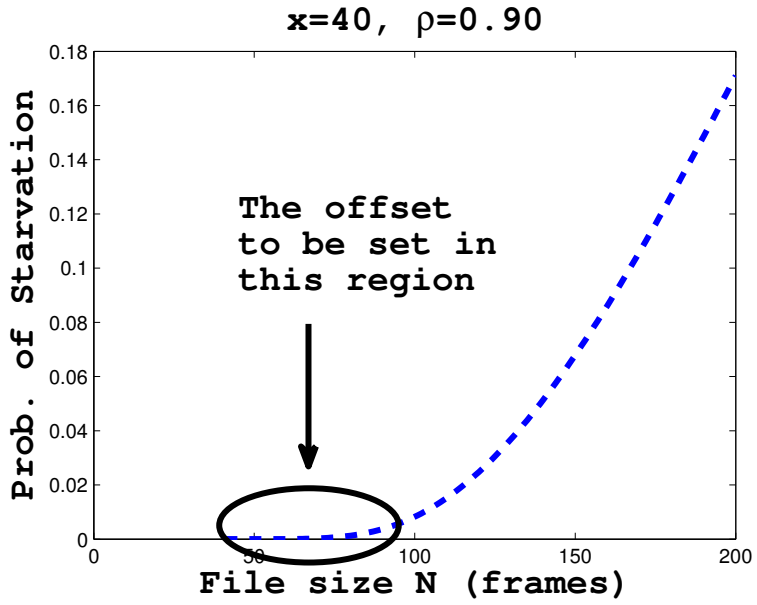

FIGURE 4.8: The offset ϕ.

where $E[T_x]$ is the expected initial buffering delay, P_s is the probability of starvation, r is the number of available bitrates, w_i is a weight associated to each bitrate and T_i is the fraction of time spent in each bitrate level. We use this function to evaluate the BSC scheme performance on the QoE metrics, i.e., how this scheme can evaluate the objective QoE cost $C(x, N)$. The examples are shown in section 4.3.2.

4.3 Simulations and numerical results

4.3.1 Simulation and validation

The mathematical models of QoE metrics are computed and compared with event-driven simulations using Matlab. A timer generates random variable stamps that record the frames arrival and departure. We monitor the playout buffer length based on the frames arrival rate and the playback rate. A quality switching occurs when there is no more frames of high quality. In this case, the player displays only the low quality frames until high quality frames are available again. A starvation happens when the buffer is empty. We vary the parameters: frames arrival rate λ, service rate μ, the file size N, the startup delay threshold x and the offset ϕ. For a given values of these parameters, we run each simulation for 4000 times.

Fig. 4.9 shows the probability of starvation given the file size N, for different settings of the traffic load ρ. The start-up threshold $x = 40$ while the offset $\phi = 50$. The probability of starvation decreases when the network throughput increases. Obviously, the video streaming users only suffer from the playback interruption when $\rho < 1$.

Fig. 4.10 shows the impact of the startup threshold x on the probability of starvation. It confirms the previous works on the tradeoff between the startup delay and the playback interruptions [45]. Indeed, we decrease a lot the video playback interruptions by accumulating more frames (also known as the prefetching process) at the beginning of the video session. The two extreme cases are: prefetching of the entire video file and no prefetching. In the former case, there is no playback interruption since all the

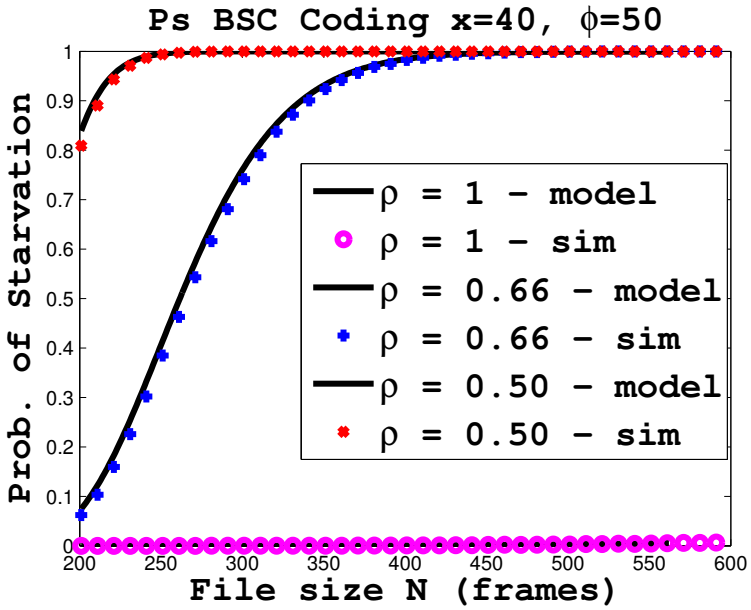

FIGURE 4.9: The probability of starvation vs the file size N

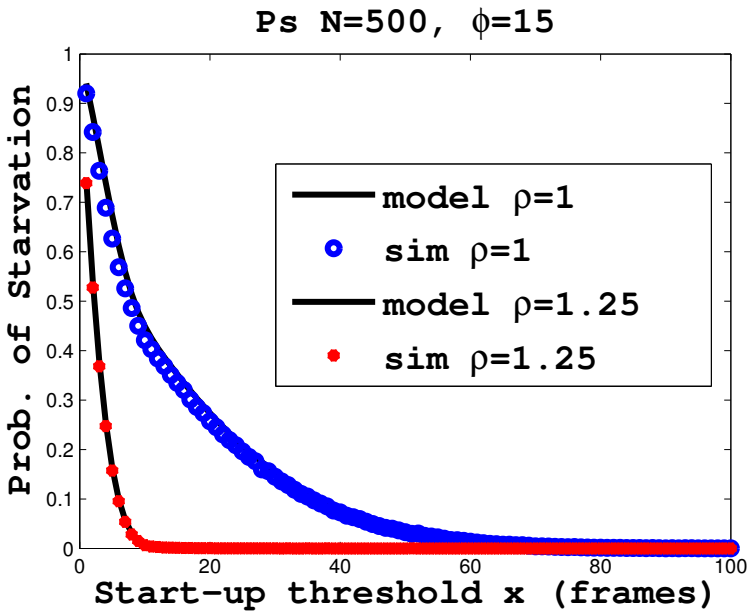

FIGURE 4.10: The probability of starvation vs the startup threshold x

video frames are in the buffer before start playing. Without prefetching, the starvation happens for sure.

We further evaluate the probabilities of having no starvation, at least one and two starvations, given the file size N for $\rho = 0.66$. Fig. 4.11 shows that the analytical model predicts the starvation probabilities accurately. The probability of no starvation decreases from 1 to 0, while the probability of having at least one and two starvations increases. The lag between the two curves gives an idea about the mean playback time, i.e., the mean time between two consecutive starvations. Then, based on the file size, the startup delay, the offset and the traffic load, the model predicts the number of

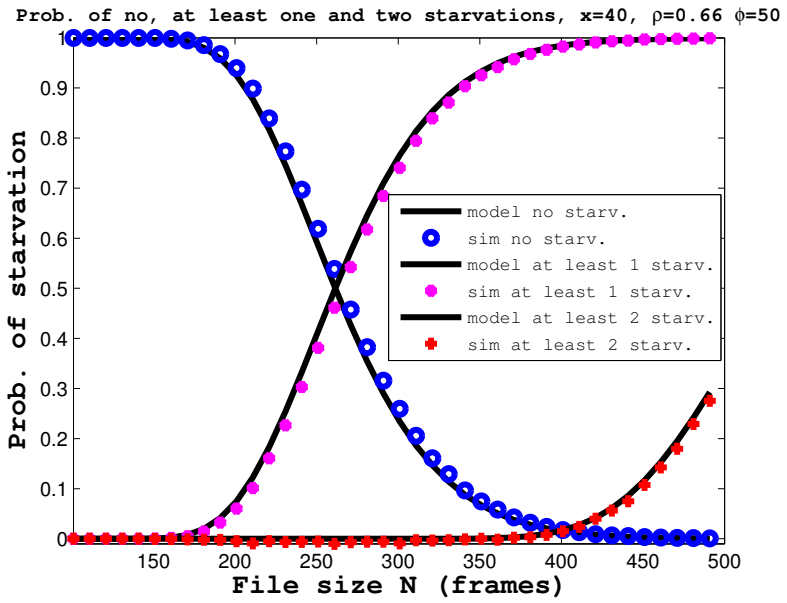

FIGURE 4.11: The probability of no starvation, or having at least one and two starvations vs the file size N

starvations that could happen during the video session.

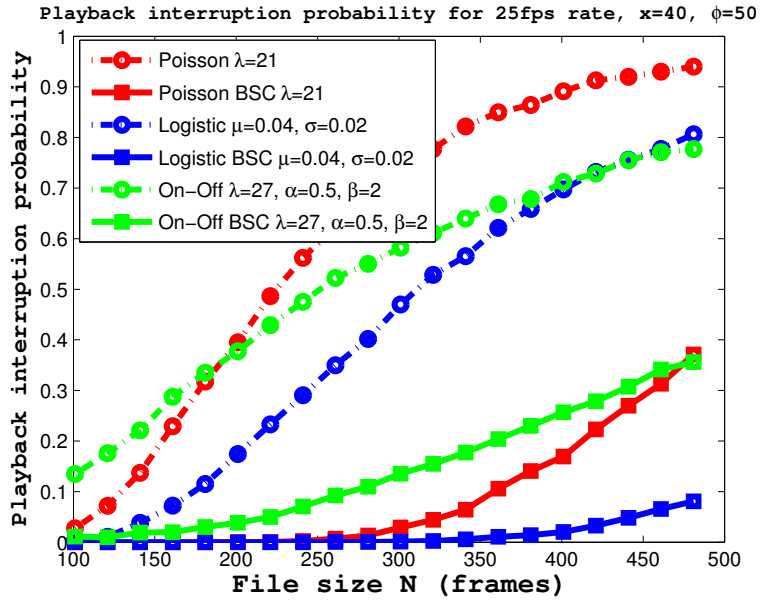

FIGURE 4.12: The playback interruption probability using several packet arrivals processes

We also evaluate BSC scheme under different types of arrival processes: the logistic distribution and the ON/OFF arrival process. In Fig. 4.12 we show that BSC scheme reduces the playback interruption. Furthermore, we obtain an important improvement of the probability of playback interruption where the arrival of frames follows the logistic process.

4.3.2 QoE optimization

We consider five video resolutions (1080p, 720p, 480p, 360p and 240p) with the corresponding bitrates (4500Kbps, 2500Kbps, 1000Kbps, 750Kbps and 400Kbps). The network throughput is 2200Kbps or 3000Kbps. Then we compute the corresponding traffic loads ρ. We compare the QoE metrics between a simple system based SVC and the BSC system. The BSC system is based on the SVC codec, then, we compare it to SVC since we know that SVC adds about 10% encoding overhead compared with the same quality AVC (the comparison between BSC and AVC is not simple without a real implementation). This comparison is done on a video file with a single quality, i.e., we do not assume quality adaptation. The quality adaptation in BSC system is studied in the next chapter. We will show the two appropriate bitrates to select for each segment, and compare the video quality to the classical DASH system.

Let assume that the network throughput is 2200Kbps. In non BSC system, we select the bitrate that is just under the network throughput, i.e., 480p resolution, in order to avoid severe playback interruptions. Since BSC superposes the base layer and the enhancement layer of two different frames, what happens if we select 480p+360p, 720p+360p or 720p+480p?

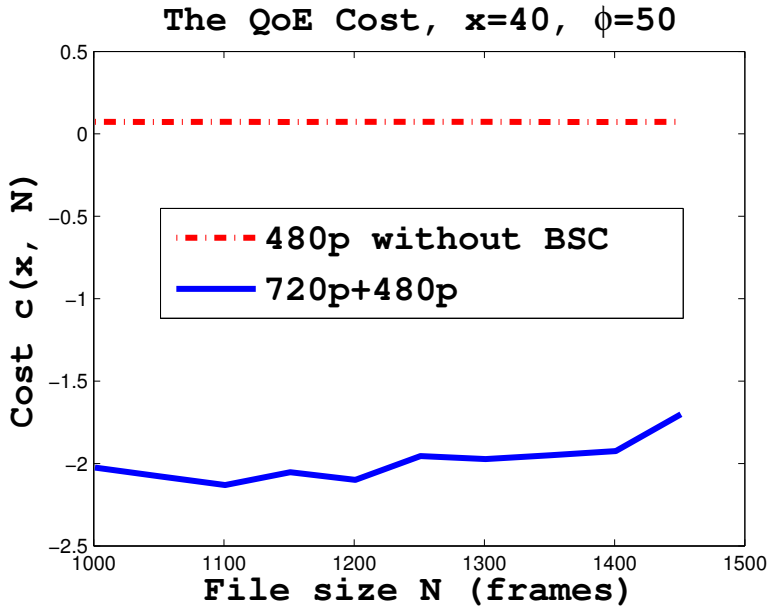

FIGURE 4.13: The QoE cost function for 480p and 720p+480p, $\gamma_1 = 0.1$, $\gamma_2 = 1$, $\gamma_3 = 0.01$

When we select 480p+360p in BSC, the cost of non BSC system is better than BSC although there is no starvation in both cases. Indeed, the non BSC system just benefits from a relatively small startup delay. However, selecting the resolutions 720p+360p or 720p+480p allows to minimize the QoE cost function as shown in Fig. 4.13. There is only one starvation for a file size of 1500 frames but with a better rendering quality. In Fig. 4.14, we show the probability of starvation for non BSC system with 720p and BSC system with 720p+480p. Notice that the high quality in both cases is 720p. We decrease a lot the probability of starvation in the case of BSC. Therefore, we can increase the probability of starvation in BSC system by increasing the quality. That is exactly what we show in Fig. 4.13. This is an important result for the integration of BSC in DASH since with the quality adaptation, we always avoid unnecessary playback

interruptions. Thus, we can increase the video quality. When the network throughput changes to 3000Kbps, BSC can use 1080p video resolution while the non BSC cannot without playback interruptions.

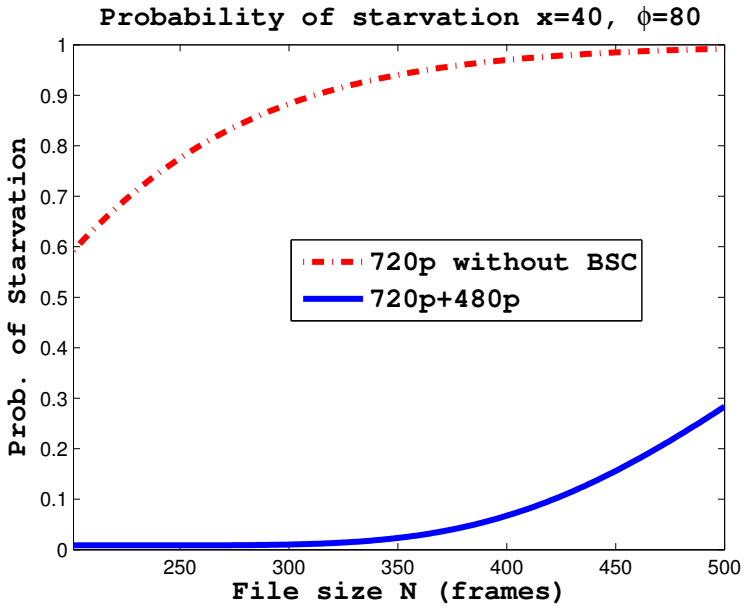

FIGURE 4.14: The probability of starvation for 720p and 720p+480p

4.4 Summary

We analyse the performance of the Backward-Shifted Coding (BSC) scheme and compare it with simple streaming system based on SVC codec.

For this purpose, we provide an analytical characterization of the dominating QoE factors: startup delay, playback buffer starvation and average video quality. We compute the first two metrics using the Ballot theorem and obtain explicit results for the probability generation function of the buffer starvation. The average video quality is computed using the quasi-stationary approach after modelling the frames arrival and departure processes as a continuous time birth-death process. Finally, we show by using a QoE objective function that BSC improves both the video playback interruptions and the average quality of the video.

In the next chapter, we will perform the integration of BSC in Dynamic Adaptive Streaming over HTTP and propose novel quality adaptation algorithms.

Chapter 5

Bitrate Adaptation in BSC for HTTP Adaptive Video Streaming

We study bitrate adaptation algorithms based on the Backward-Shifted Coding (BSC) scheme introduced beforehand in Chapter 3. As mentioned in this chapter, the video sequence is divided in smaller chunks called segments, each one in multiple layers (base layer and enhancement layers). The segments layers transmission with the Backward-Shifted Coding scheme consists on sending each segment twice: a low layer segment containing a base layer and possibly some enhancement layers and a top layer segment containing only enhancement layers. The superposition of the low layer and the top layer of two different segments is called a BSC block. The purpose of this chapter is the selection of the quality (bitrate) of the low layer segment and the top layer segment within each BSC block given the network capacity in order to enhance the quality of experience of the users: minimize the video playback interruptions, increase the video quality and decrease the video quality variations. We first describe the segments level system with BSC. Then, we propose bitrate adaptation algorithms in BSC system able to balance the need for maximum quality, video rate smoothness while avoiding the risk of playback interruptions. Finally, we provide simulations using synthetic and real-world video traffic traces to show the benefits of the Backward-Shifted Coding system.

5.1 System description

We consider an HTTP adaptive video streaming system in which the server holds the Media Presentation Description (MPD), the media (audio and video) segments and it hosts a HTTP server. The information related to the media segments, e.g., the available video bitrates, are described in the MPD document. In the Backward-Shifted Coding, the video segments are transmitted within BSC blocks. As shown in Fig. 5.1, block k contains segment $k + \phi - 1$ (lower layer segment) and enhancement layers of segment k (top layer segment) where ϕ is the offset between the low layer segment and its corresponding top layer segment. Since the top layer segment contains only enhancement layers, we use the terms "top layer segment" and "enhancement layer segment" interchangeably in this chapter.

Each time a user requests the video, a HTTP connection is established with the server. The MPD document is sent first before any video data is transmitted. The video blocks are downloaded into a playback buffer, which contains downloaded segments but are not yet displayed by the playout. As shown in Fig. 5.1, after block k is downloaded, segment k can be decoded using the low layer segment from block $k - \phi + 1$ and the enhancement layers from block k. Note that the index of block k refers to the index of the segment to which the top layer belongs to.

Let K be the number of segments contained in the video file. Each segment contains T_s seconds of video and it is encoded at different bitrates. In standard SVC playout, a

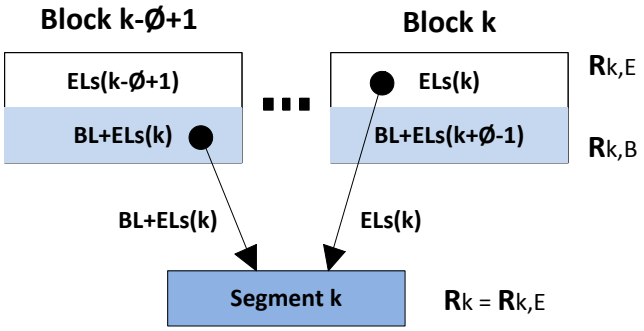

FIGURE 5.1: Decoding of segment k: uses low layer segment of block $k-\phi+1$ (containing base layer and possibly some enhancement layers) and top layer segment of block k (containing enhancement layers only).

set of available bitrate levels per segment $\mathcal{R}$ corresponds to selecting the base layer and a certain number of enhancement layers. In the BSC system, the playback downloads the BSC block k with the bitrates $(R_{k,E}, R_{k,B}) \in \mathcal{R}^2$. In particular we denote:

- $R_{k,E}$ is the bitrate of segment k by including the low layer segment, which is received through block $k-\phi+1$
- $R_{k,B}$ is the bitrate of the low layer segment $k+\phi-1$ (which contains base layer and some enhancement layers).

Note that, with this notation, when we refer to the condition $R_{k,E} = R_{k-\phi+1,B}$, we mean that no enhancement layers are transmitted in block k. The bitrate notation for all the segments is represented in Fig. 5.2.

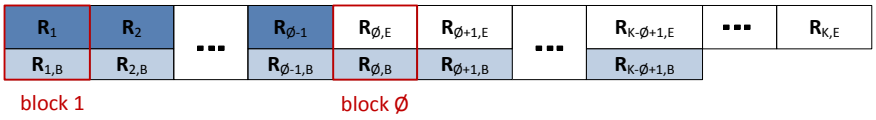

FIGURE 5.2: The low layers and top layers segments bitrates notations: In the notation $R_{k,B}$ or $R_{k,E}$, k refers to the index of the block. It is also the index of the top layer segment. But k is not the index of the low layer segment: for example, in the first block, $R_{1,B}$ is the bitrate of the first low layer segment which is segment ϕ.

Let $d_{k,E}$ and $d_{k,B}$ be, respectively, the size of the enhancement layers segment and the size of the low layer segment in block k. Thus

$$d_{k,E} = (R_{k,E} - R_{k-\phi+1,B})T_s, \text{ and } d_{k,B} = R_{k,B}T_s$$

so that the corresponding rate for block k given by $R_k = R_{k,E} - R_{k-\phi+1,B} + R_{k,B}$. Clear, the set of all possible block bitrates is still $\mathcal{R}$.

5.2 Adaptation methods in BSC

The goal of the bitrate adaptation is to maximize the QoE of the video streaming users depending on four key parameters: the startup delay, the playback interruption, the mean video bitrate and the bitrate switching or quality variations. Herein we

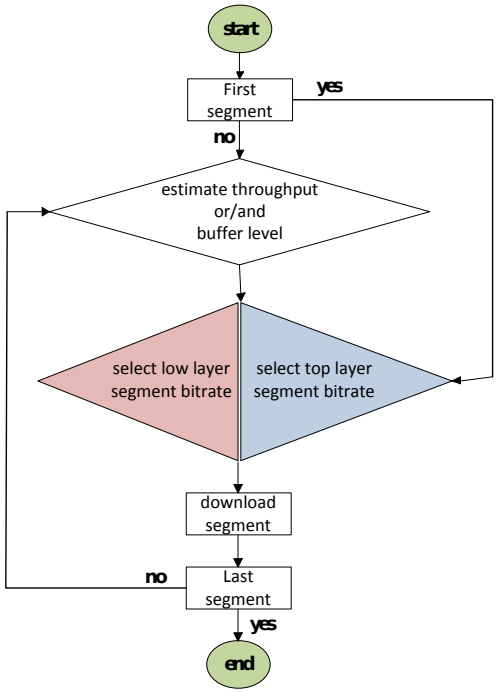

FIGURE 5.3: Flow diagram for throughput-based and buffer-based adaptation algorithms for BSC.

propose bitrate adaptation methods to choose the suitable bitrates for block k (Fig. 5.3).

In order to select bitrates $R_{k,B}$ and $R_{k,E}$, we are inspired from the approaches from the literature, namely the buffer-based and the throughput-based approach as described beforehand in Chapter 2. This results into two algorithms: the throughput-based BSC algorithm (TB-BSC) and the buffer-based BSC algorithm (BB-BSC). The main idea behind throughput-based schemes is that the MPEG-DASH client performs an estimation of the available bandwidth for the requested segments [101, 102]. Then, based on the network throughput and the playout buffer occupancy level, an adaptation engine chooses the highest possible bitrate compatible with the available throughput in order to avoid possible playback interruptions. The simplest way to estimate the available throughput is to compute the segment throughput after it is completely downloaded. This is a standard throughput measure called *instant throughput* [60]. This method is simple but not accurate due to the estimation errors. The estimated throughput can be generated based on other techniques such as exponential average or weighted average in order to mitigate short-term fluctuations caused by the low layers.

Conversely, buffer-based methods leverage on the size of the buffer, with the aim of keeping it at a given nominal level.

5.2.1 The Throughput Based BSC Algorithm: TB-BSC

We denote by R_{min} and R_{max} the smallest and the highest bitrate respectively in the set of available bitrates $\mathcal{R}$. We let $\hat{A}_t$ and $\mathcal{B}_k$ be, respectively, the estimated throughput after the download of the segment $k-1$ and the current playback buffer occupancy measured in seconds of video content. We distinguish two cases based on the block index: $k < \phi$ and $k \geq \phi$.

Case $0 \leq k \leq \phi - 1$**.** For the $\phi - 1$ first blocks, see Fig. 5.4, each block contains 1) the low layer segment k (upper segment) and 2) the low layer segment $k + \phi - 1$ but at minimum bitrate $R_{k,B} = R_{min}$. Thus, for the first $\phi - 1$ blocks, the bitrate adaptation concerns only the low layer segment k and must be operated such in a way that $R_k + R_{min} \leq \hat{A}_t$ where R_k is the bitrate of the low layer segment k.

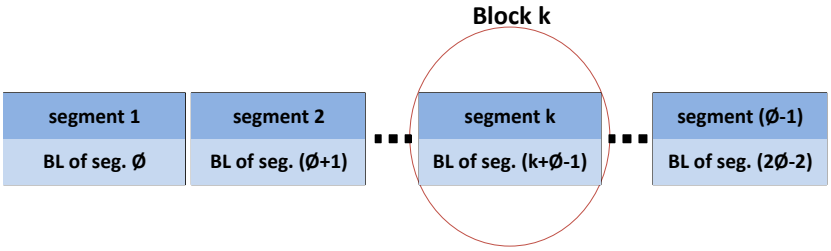

FIGURE 5.4: Block k contains segment k and BL of segment $k + \phi - 1$ for $k < \phi$

By assigning a minimum bitrate, R_{min}, to the low layer segment $k + \phi - 1$, the startup delay is not greatly affected by the BSC scheme. Therefore, we immediately maximize the bitrate of the segments $1 \leq k \leq \phi - 1$ – for which no enhancement layers are expected later on – and we defer the bitrate enhancement of the low layer segments $\phi \leq k \leq 2\phi - 2$ using the top layer segment carried by block $k + \phi - 1$.

Case $k \geq \phi$**.** The pseudo-code for this part of the TB-BSC adaptation algorithm is provided in Alg. 1. It is interesting to observe that, in TB-BSC scheme, we shall also leverage on information on the buffer level occupancy.

We assume that the algorithm is invoked repeatedly each time t a block is downloaded; it starts immediately after the download of BSC block $k-1$ is completed.

Let $\phi_t = \phi \cdot T_s$: it represents the offset in seconds between the low layer segment and its enhancement layers (top layer segment). When the buffer size (in seconds) is not larger than ϕ_t (Line 3), we no longer need to send the enhancement layers segments because their corresponding segments are already been played by the playout. In that case, the bitrate selection is equivalent to DASH based SVC solutions (Line 8 to 16). When $\mathcal{B}_k > \phi_t$ (Line 18 on), the adaptation is done on both the low layer segments and the enhancement layers segments.

In the worst case, i.e. Line 19, when the estimated throughput is lower than R_{min}, the selected bitrate for the low layer segment in block k is R_{min} and no enhancement layers are sent, i.e., $R_{k,E} = R_{k-\phi+1,B}$.

We denote by R_{t-}, the highest available bitrate compatible with the estimated throughput. In the same way, R_{t+} is the smallest available bitrate regarding the estimated throughput.

$$\begin{cases} R_{t-} &= \{\max\{R_i\} : R_i \leq \hat{A}_t\} \\ R_{t+} &= \{\min\{R_i\} : R_i \geq \hat{A}_t\} \end{cases}$$

When the estimated throughput is lower than the bitrate of the low layer segment in

```
   Input:
   Â_t: the estimated throughput of block k − 1
   R_{k−1,B}: the bitrate of the low layer segment in block k − 1
   R_{k−φ+1,B}: the bitrate of the low layer segment in block k − φ + 1
   B_k: the buffer occupancy in seconds
   Output:
   R_{k,B}: the bitrate of the low layer segment in block k
   R_{k,E}: the bitrate of the enhancement layer segment in block k
 1 R_{t−} ← {max(R_i) : R_i ≤ Â_t} ;
 2 R_{t+} ← {min(R_i) : R_i ≥ Â_t} ;
 3 if B_k ≤ φ_t then
 4 |   R_{k,E} := R_{k−φ+1,B} //no enhancement layers;
 5 |   if Â_t ≤ R_min then
 6 |   |   R_{k,B} := R_min ;
 7 |   else
 8 |   |   if Â_t < R_{k−1,B} then
 9 |   |   |   R_{k,B} := R_{t−} ;
10 |   |   else
11 |   |   |   if R_{k−1,B} < R_max then
12 |   |   |   |   R_{k,B} := R↑_{k−1,B} ;
13 |   |   |   else
14 |   |   |   |   R_{k,B} := R_max ;
15 |   |   |   end
16 |   |   end
17 |   end
18 else
19 |   if Â_t ≤ R_min then
20 |   |   R_{k,B} := R_min ;
21 |   |   R_{k,E} := R_{k−φ+1,B} ;
22 |   else
23 |   |   if Â_t < R_{k−1,B} then
24 |   |   |   R_{k,B} := R_{t−} ;
25 |   |   |   R_{k,E} := max(R_{k−φ+1,B}, R_{t+}) ;
26 |   |   else
27 |   |   |   if R_{k−1,B} < R_max then
28 |   |   |   |   R_{k,B} := R↑_{k−1,B} ;
29 |   |   |   |   R_{k,E} := max(R_{k−φ+1,B}, R↑_{k−1,E}) ;
30 |   |   |   else
31 |   |   |   |   R_{k,B} := R_max ;
32 |   |   |   |   R_{k,E} := R_max ;
33 |   |   |   end
34 |   |   end
35 |   end
36 end
```

Algorithm 1: TB-BSC Algorithm for $k \geq \phi$

the previous block $k-1$, the bitrate of the low layer segment in the next block k is set to R_{t-}. And the bitrate of the enhancement layers segment in the next block k is the maximum between R_{t+} and $R_{k-\phi+1,B}$ (Line 24 and 25, respectively). It is worth remarking that in this case, the selected bitrate for the low layer segment is not larger

than the estimated throughput in order to prevent playback interruptions. But, we observe that the bitrate of the enhancement layers segment is larger than the estimated throughput. Indeed, we do not risk playback interruptions here: in fact the buffer level is larger enough ($\mathcal{B}_k > \phi_t$).

When the available throughput increases compared to the previous block (Line 26), we increase the bitrate in a smooth manner in order to avoid sudden video quality transitions [103]. In practice, when the estimated throughput is higher than the bitrate of the low layer segment in the block $k-1$, the selected bitrate of the low layer segment in the block k is increased to a higher bitrate, i.e., $R_{k,B} = R^{\uparrow}_{k-1,B}$, (Line 28). The bitrate of the enhancement layers of block k is increased to a higher bitrate as well (Line 29). Note that, when $R_{k,E} > R_{k-\phi+1,B}$, the necessary number of enhancement

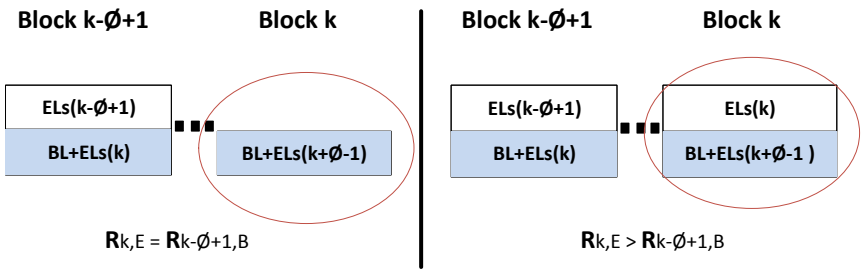

FIGURE 5.5: If $R_{k,E} = R_{k-\phi+1,B}$, no enhancement layers are transmitted in block k, otherwise, the necessary number of enhancement layers are added to block k to reach bitrate $R_{k,E}$

layers are added to the segment in block k to reach the bitrate $R_{k,E}$. This is illustrated in Fig. 5.5.

Finally, in the Alg.1, the bitrate of the lower layer segment in block k is R_{t-} and the bitrate of the enhancement layers segment is R_{t+}. However, when $R_{k,B} = R_{t-}$ and $R_{k,E} = R_{t-}$, the Backward-Shifted Coding system is equivalent to the DASH/SVC system.

5.2.2 The Buffer Based BSC Algorithm: BB-BSC

The use of buffer occupancy to select the segments' bitrate is a technique used by several schemes in the literature [61, 104, 55] as mentioned beforehand in Chapter 2. Typically, buffer thresholds are set (either two or three thesholds) and decisions on the bitrate are taken according to the level of current buffer occupancy with respect to such thresholds. Some of these methods use also the estimated throughput to smooth bitrate variations. Let us call BBA-0 this group of bitrate adaptation methods.

However, there exist another group of buffer-based algorithms where an adjustment function is used to pick the appropriate bitrate [64, 65]. Let us call them BBA-1: compared to BBA-0, they do not perform throughput estimation, thus avoiding the related estimation errors. We use BBA-0 and BBA-1 to design our BSC buffer based algorithms. BBA-1 is the basis of BB-BSC-1 algorithm where BBA-0 is the basis of BB-BSC-0 algorithm. We first describe the application to BSC of the template algorithm introduced in [64], shortly BBA-1. Then we specialize it to match the specific features of BSC and derive BB-BSC-1. BB-BSC-1 will be finally composed of two procedures, one for the low layer segments and one for the top layer segments. Those are reported in Alg. 2

and Alg. 3, respectively. The purpose of the low layer segments algorithm is to increase the video quality while the top layer segments algorithm enhances both the video quality and the quality variations. Thereafter we explain the design of BB-BSC-0 algorithm.

BB-BSC-1

Low layer segments algorithm:

We have two buffer thresholds r and c where r is the reservoir and c is the cushion in seconds of video content. The bitrate selection is based on an adjustment function F where $F(\mathcal{B}_k) = R_{min}$ for $\mathcal{B}_k \leq r$ and $F(\mathcal{B}_k) = R_{max}$ for $\mathcal{B}_k \geq r + c$. Then, given the current buffer occupancy $\mathcal{B}_k$, $F(\mathcal{B}_k)$ is computed to select the bitrate of the next segment. We use the following function F as in [58]:

$$F(\mathcal{B}_k) = \begin{cases} R_{min} & \mathcal{B}_k \leq r \\ R_{max} & \mathcal{B}_k \geq r + c \\ R_{min} + \frac{\mathcal{B}_k - r}{c}(R_{max} - R_{min}) & otherwise \end{cases} \tag{5.1}$$

First, we remark that when using BBA-1 algorithm on the low layer segments in BSC system with the adjustment function F, we still have a margin which can be used to add enhancement layers segments while avoiding the playback interruptions. Therefore we define two adjustment functions F_1 and F_2. The two functions have the same formula as function F but differ in the value of c, i.e, F_1 uses c_1 and F_2 uses c_2 (see Fig. 5.6). Given the values of c_1 and c_2, we can increase and decrease the margin between the two

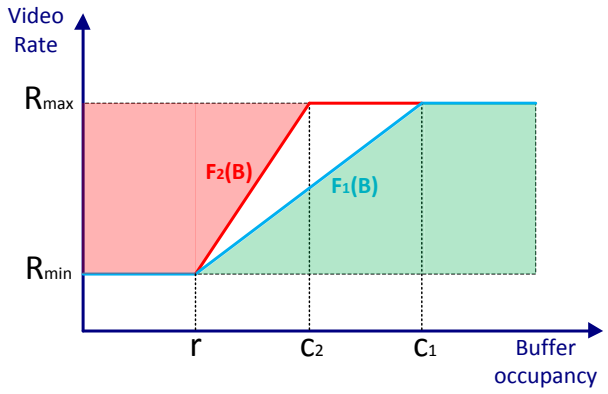

FIGURE 5.6: The adjusment functions for the low layer segments and the top layer segments: rates above the curve are risky for buffer starvation, rates below the curve are safer but correspond to lower quality.

curves and then adjust the amount of enhancement layers segments we add to the low layer ones.

The inputs of the algorithm are the bitrate of the previous low layer segment, the current buffer occupancy and the buffer thresholds. The output is the bitrate of the next low layer segment. Then, we compute the bounds of the previous bitrate (R_+ and R_-) and the adjustment function F_1 regarding the buffer occupancy $\mathcal{B}_k$. The bitrate of the next low layer segment is selected according to $F_1(\mathcal{B}_k)$ and the buffer occupancy.

Input:
$R_{k-1,B}$: the bitrate of the low layer segment in block $k-1$
$\mathcal{B}_k$: the current buffer occupancy
r and c_1: the sizes of the reservoir and the cushion
Output:
$R_{k,B}$: the bitrate of the low layer segment in block k

if $R_{k-1,B} = R_{max}$ **then**
  $R_+ = R_{max}$
else
  $R_+ = min\{R_i : R_i > R_{k-1,B}\}$
end
if $R_{k-1,B} = R_{min}$ **then**
  $R_- = R_{min}$
else
  $R_- = max\{R_i : R_i < R_{k-1,B}\}$
end
if $\mathcal{B}_k \leq r$ **then**
  $R_{k,B} = R_{min}$
else if $\mathcal{B}_k \geq r + c_1$ **then**
  $R_{k,B} = R_{max}$
else if $F_1(\mathcal{B}_k) \geq R_+$ **then**
  $R_{k,B} = max\{R_i : R_i < F_1(\mathcal{B}_k)\}$
else if $F_1(\mathcal{B}_k) \leq R_-$ **then**
  $R_{k,B} = min\{R_i : R_i > F_1(\mathcal{B}_k)\}$
else
  $R_{k,B} = R_{k-1,B}$
end
return $R_{k,B}$

Algorithm 2: low layer segment algorithm

Top layer segments algorithm: Smoothing the bitrate variability

The main purpose of the enhancement layers segments is to improve the quality of the video. They do not increase the video content in the buffer in terms of playout time. For the first $\phi - 1$ blocks, the buffer level increases by two segments after the download of block k. Whereas for $k \geq \phi$, the buffer level increases by only one segment after the download of block k is completed. We use the adjustment function F_2 to select the bitrate of the enhancement layers segments. Since $F_2 \geq F_1$, we will increase the video quality. But we also want to decrease the quality variations. For this purpose, we will apply the Alg. 3 not on a single enhancement layer segment, but on a set of blocks of enhancement layers segments of length $\phi - 1$ in order to smooth the bitrate.

An example of this smoothing procedure is reported in Fig. 5.7 for $\phi = 4$. The algorithm is applied on blocks of 3 consecutive enhancement layers segments. The red part represents the low layer segments. After the download of top layer segment 3, the output of the algorithm is R_i (the green bar). Then, we have to download the necessary enhancement layers of segments 4, 5 and 6 to reach R_i. These enhancement layers will be download on low layer segments 7, 8 and 9 respectively.

The algorithm is invoked after a set of blocks of length $\phi - 1$. Then, when the algorithm is invoked after the download of block $k-1$, the output remains the same for the next $\phi - 1$ BSC blocks (mod $(k-1, \phi-1)$). The inputs are the previous enhancement layers segment bitrate, the buffer occupancy and the buffer thresholds r and c_2. The output is the bitrate of the next $\phi - 1$ enhancement layers segments. For the algorithm of the low layer segments, we compare $F_1(\mathcal{B}_k)$ to the bounds of the bitrate of the previous low layer segment. Here, we compare $F_2(\mathcal{B}_k)$ to r^+_{avg} and r^-_{avg}. r^+_{avg} (r^-_{avg}) is the bitrate of each segment in the the previous set of blocks of length $\phi - 1$

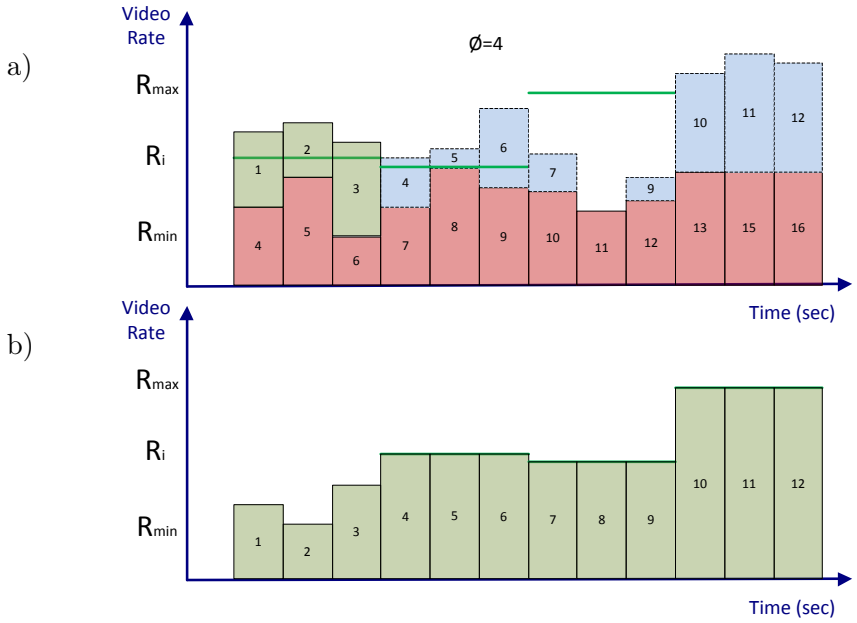

FIGURE 5.7: a) Example of application of Alg. 3: applied on several segments simultaneously it smooths the bitrate variability b) Effect at the decoder side; $\phi = 4$.

to reach R_+ (R_-) where R_+ (R_-) is the upper (lower) bound of the previous bitrate $R_{k-1,E}$. In other words, $r^+_{avg} = R_+$ and $r^-_{avg} = R_-$. Then, we compute the bounds of the previous bitrate and the adjustment function F_2 corresponding to buffer occupancy $\mathcal{B}_k$. The bitrate of the next enhancement layers segment is selected according to $F_2(\mathcal{B}_k)$ and the buffer occupancy.

BB-BSC-0

Compared to the previous algorithm BB-BSC-1, BB-BSC-0 algorithm does not use an adjustment function. It only uses the buffer thresholds to decide on the bitrate of the segment. Therefore, we define three buffer thresholds B_1, B_2 and B_3 in seconds of video content. The goal of this buffer based approch is to keep the buffer occupancy level $\mathcal{B}_k$ between B_{low} and B_{high}.

As for the previous algorithms, when $\mathcal{B}_k \leq \phi_t$, the top layer segments are not requested. Then, the bitrate selection is made only on the low layer segments. Otherwise, the bitrate decision is made on both the low layer and the top layer segments.

When the buffer occupancy level is under B_1, the bitrate of the low layer segment is R_{min} and no top layer segment is requested. When the buffer occupancy level is under B_2, the bitrates of both low and top layer segments are decreased. When the buffer occupancy level is between B_2 and B_3, we do not change the bitrate of the low layer and the top layer segments. Finally, when the buffer occupancy level is higher than B_3, the bitrates of the low layer and the top layer segments are increased.

We observe that the video playback stays at the lowest bitrate for a long time ($\approx B_3$ seconds). So we need to quickly increase the video quality at the beginning of the video playback. For the beginning phase, we use the throughput based algorithm 1. This phase terminates when we reach the highest available bitrate R_{max} or when the selected bitrate is close to the estimated throughput.

Input:
$R_{k-1,E}$: the bitrate of the enhancement layers segments of the previous $\phi - 1$ blocks
$\mathcal{B}_k$: the current buffer occupancy
r and c_2: the sizes of the reservoir and the cushion
Output:
$R_{k,E}$: the bitrate of the enhancement layers segments of the next $\phi - 1$ blocks

if mod $(k-1, \phi-1) == 0$ **then**
  if $R_{k-1,E} = R_{max}$ **then**
    $R_+ = R_{max}$
  else
    $R_+ = min\{R_i : R_i > R_{k-1,E}\}$
  end
  if $R_{k-1,E} = R_{min}$ **then**
    $R_- = R_{min}$
  else
    $R_- = max\{R_i : R_i < R_{k-1,E}\}$
  end
  $r^+_{avg} \leftarrow [\sum_{i=k-2\phi+2}^{k-\phi} R_{i,B} + max(0, (R_+ - R_{i,B}))]/(\phi - 1)$
  $r^-_{avg} \leftarrow [\sum_{i=k-2\phi+2}^{k-\phi} R_{i,B} + max(0, (R_- - R_{i,B}))]/(\phi - 1)$
  if $\mathcal{B}_k \leq r$ **then**
    $R_{k,E} = R_{k-\phi+1,B}$ //no enhancement layers
  else if $\mathcal{B}_k \geq r + c_2$ **then**
    $R_{k,E} = R_{max}$
  else if $F_2(\mathcal{B}_k) \geq r^+_{avg}$ **then**
    $R_{k,E} = max\{R_i : R_i < F_2(\mathcal{B}_k)\}$
  else if $F_2(\mathcal{B}_k) \leq r^-_{avg}$ **then**
    $R_{k,E} = min\{R_i : R_i > F_2(\mathcal{B}_k)\}$
  else
    $R_{k,E} = R_{k-1,E}$
  end
  return $R_{k,E}$
else
  return $R_{k-1,E}$
end

Algorithm 3: top layer segment algorithm

5.3 Simulations and numerical results

5.3.1 Simulation setup

We evaluate the bitrate adaptation in TB-BSC and BB-BSC using a custom simulation framework. The simulation takes as inputs the network capacity, the number of segments in the video file K, the segment duration T_s, the offset ϕ and the set of available bitrates $\mathcal{R}$. Then, we model the video downloading, the playback process and the buffer dynamics. The startup delay corresponds to the download time of the first BSC block. After the end of the startup phase, the playback starts with a display speed of d frames per second (fps). We compute the size of each block k according to the bitrates $R_{k,B}$ and $R_{k,E}$. Then the block is downloaded using the real network capacity. At the end of the download of block k, we select the bitrates of the next block $k+1$. At the user side, the block is decoded once it is completely downloaded and the segments are available in the buffer.

We test the algorithms of the previous sections under different network capacity variations. The network capacity of Fig. 5.8 is a generated network traffic model. It is a network capacity model similar to that of [57]: it is a realistic network bandwidth variations with different congestion states, where background TCP traffic is injected between the server and the client. We set the link capacity between the server and

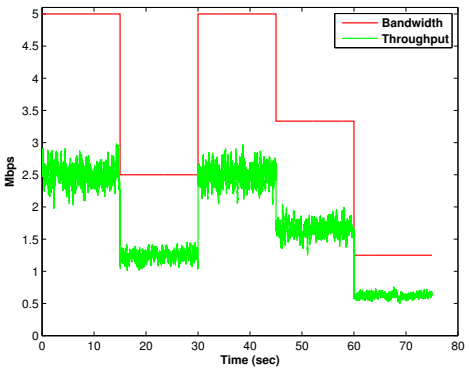

FIGURE 5.8: Background traffic with state variations

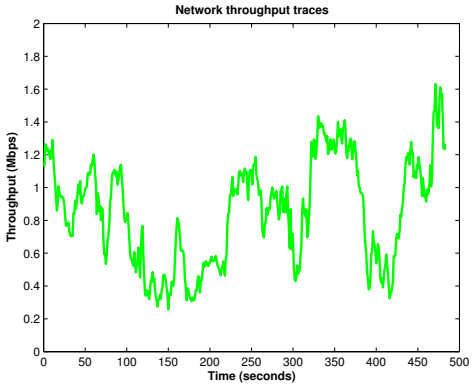

FIGURE 5.9: Network throughput trace from 3G/HSDPA mobile wireless network

the client to be 5Mbps. We simulate the behaviour of a standard linux *traffic control tool* by decreasing or increasing the throughput. The traffic rate thus jumps between different states. The oscillations within the same state are due to TCP congestion control mechanisms. The link capacity is higher than the TCP throughput: we assume that this throughput is used only for the video segments transmission since the audio segments and the synchronisation data can still use a fraction of the whole link capacity.

We also consider a real HSDPA trace reporting logs from TCP streaming sessions in Telenor's 3G/HSDPA mobile wireless network in Norway [105](Fig. 5.9). It is an average throughput trace collected from several measurements inside a tramway from Ljabru (start location) to Jernbanetorget (destination location). For each measurement, adaptive video streams were downloaded at the maximum speed with a video segment duration of 2 seconds. We summarize the algorithms notations in Table 5.1.

TB-SVC	throughput based algorithm of DASH using SVC: it is the classic DASH algorithm where the selected bitrate is always lower than the estimated throughput
TB-BSC	throughput based algorithm of BSC: Alg. 1
BBA-0	buffer based algorithm using 3 thresholds: B_1, B_2 and B_3
BBA-1	buffer based algorithm using the adjustment function of Eq. 5.1
BB-BSC	buffer based algorithm of BSC
BB-BSC-0	BB-BSC inspired from BBA-0
BB-BSC-1	BB-BSC inspired from BBA-1: Alg. 2 and 3

TABLE 5.1: The notations of the different algorithms

5.3.2 Numerical results

The set of experiments compares the requested bitrate with TB-BSC, BB-BSC, the throughput based scheme using SVC (TB-SVC) and the buffer based scheme using SVC (BBA-0 and BBA-1) under the network conditions of Fig. 5.8 and the throughput traces

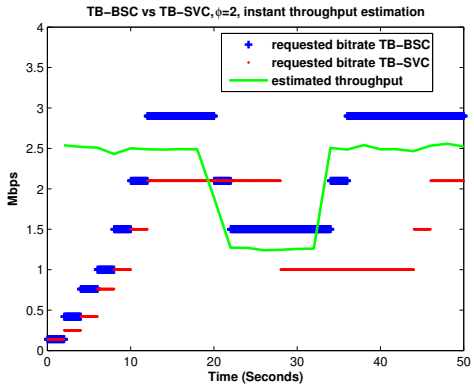

FIGURE 5.10: Requested bitrates for TB-BSC and TB-SVC for instant throughput estimation method

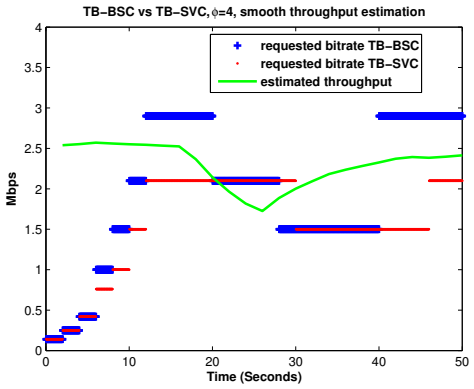

FIGURE 5.11: Requested bitrates for TB-BSC and TB-SVC for smooth throughput estimation method

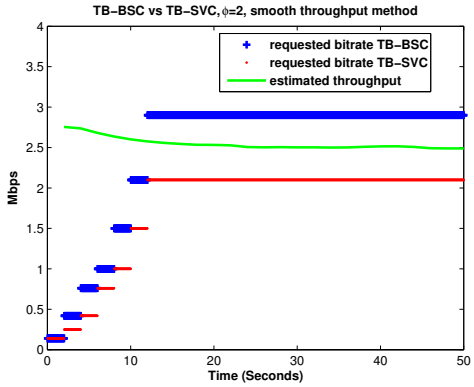

FIGURE 5.12: Requested bitrates for TB-BSC and TB-SVC with permanent bandwidth state

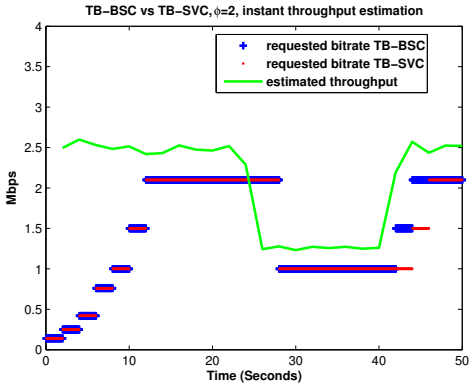

FIGURE 5.13: Requested bitrates for TB-BSC and TB-SVC

of Fig. 5.9. The file size in the experiments is up to 350 seconds (≈ 5 minutes) of video while the playback frequency is 25 frames per second (fps). We consider the following set of available bitrates { 140, 250, 420, 760, 1000, 1500, 2100, 2900 } (Kbps). The video segment duration is set to 2 seconds.

Throughput based algorithm: Fig. 5.10 shows the requested bitrate for TB-BSC and TB-SVC for the throughput based algorithm where the throughput is estimated over the last segment that was downloaded (instant throughput). TB-BSC achieves a higher average bitrate compared to TB-SVC with only one extra bitrate switching. The average bitrate is 2040 Kbps with TB-BSC and 1382 Kbps with TB-SVC. As shown in the Fig. 5.10, we begin with the smallest bitrate and we increase the bitrate in a smooth manner because according to [54], users prefer a gradual quality change. The number of quality variations is 10 for TB-BSC and 9 for TB-SVC.

Fig. 5.11 shows the requested bitrate for TB-BSC and TB-SVC for the throughput based algorithm where the throughput is estimated over all the previous segments (smooth throughput). We use the estimation method of [103] where the actual estimated throughput is 20% of the throughput estimated over the last segment and 80% of the estimated throughput over all the previous segments. The purpose of this method of estimation is to cope with the throughput short-term fluctuations. As for the instant

throughput case, TB-BSC system achieves a better video quality than TB-SVC. The average bitrate is 1948 Kbps for TB-BSC and 1566 Kbps for TB-SVC. The number of quality variations is 9 for TB-BSC and 8 for TB-SVC. If we compare these results with the instant throughput case, we see that the number of quality variations decreases. This method of throughput estimation is more robust than the instant throughput case.

We evaluate the system when the state of the bandwidth does not change (permanent bandwidth case) but it still undergoes short term fluctuations. Recall that the small throughput fluctuations are due to TCP congestion control mechanisms. We compute the requested bitrate in TB-BSC and TB-SVC in that case. Fig. 5.12 shows the results. The first observation is that BSC definitely outperforms classical DASH based SVC system since the average bitrate is 2440 Kbps for BSC against 1758 Kbps for DASH-SVC. Moreover, the two systems have the same number of quality variations (e.g., 6). The estimated throughput is around 2.5 Mbps. For the same network conditions, TB-BSC renders the video with 2900 Kbps quality while TB-SVC cannot. If we force TB-SVC to play the video with the quality of 2900 Kbps, we will have a severe number of playback interruptions.

Table 5.2 gives the average quality, the variance of the quality, the number of quality variations, and the number of the playback interruptions for 100 simulations for three scenarios: **scenario 1** is the permanent bandwidth state case with the smooth throughput estimation method while **scenario 2** and **scenario 3** are, respectively, the variable bandwidth state case with the instant and the smooth throughput estimation methods. The variance of the quality gives how far the temporal quality is from the mean quality value. Since the users prefer gradual quality variations, high value of the variance degrades the quality of experience. We observe that the Backward-Shifted

	Average quality (Kbps)	Variance of the quality	Number of switchings	Number of playback interruptions
Scenario 1 TB-BSC	**2324**	825e9	11.84	**0**
Scenario 1 TB-SVC	1758	**421e9**	**6**	**0**
Scenario 2 TB-BSC	**1514**	452e9	**9**	**0**
Scenario 2 TB-SVC	1382	**425e9**	**9**	**0**
Scenario 3 TB-BSC	**1951**	737e9	9.73	0.29
Scenario 3 TB-SVC	1567	**369e9**	**8**	**0**

TABLE 5.2: The average statistics for 100 simulations for $\phi = 4$

Coding always outperforms DASH based SVC solutions in terms of video quality, but with an additional number of quality variations and a little risk of the video playback interruptions. The buffer based methods will overcome these limitations.

The Backward-Shifted Coding system can behave exactly like TB-SVC system if we choose the same bitrate for the low layer segments $R_{k,B}$ and the enhancement layers segments $R_{k,E}$. Fig. 5.13 shows the requested bitrate for TB-BSC and TB-SVC when $R_{k,B} = R_{k,E}$. The requested bitrate is almost the same for the two systems (TB-BSC still has a higher bitrate). This short difference is due to the offset ϕ in BSC system.

This property adds more flexibility to BSC system since one can switch from TB-BSC to TB-SVC or inversely depending on the network capacity.

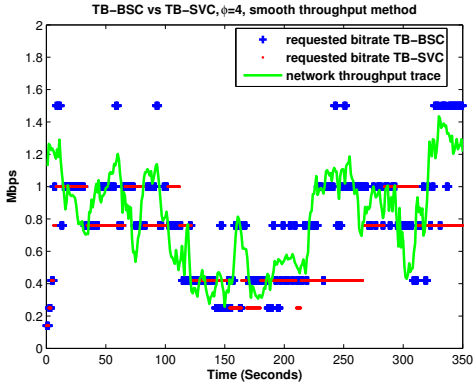

FIGURE 5.14: Requested bitrates for TB-BSC and TB-SVC for smooth throughput estimation method

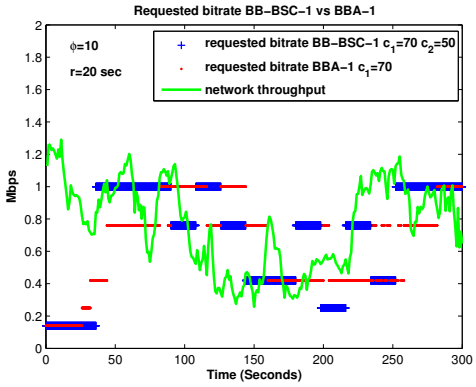

FIGURE 5.15: Requested bitrates for BB-BSC-1 and BBA-1, c_1 for BBA-1 is 70 sec

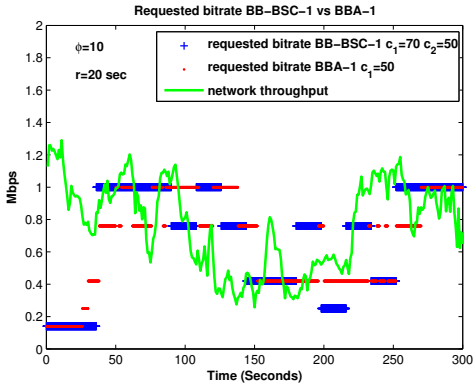

FIGURE 5.16: Requested bitrates for BB-BSC-1 and BBA-1, c_1 for BBA-1 is 50 sec

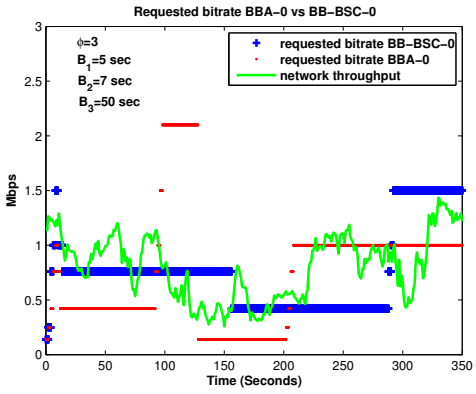

FIGURE 5.17: Requested bitrates for BBA-0 and BB-BSC-0 for $B_1 = 5$ sec, $B_2 = 7$ sec and $B_3 = 50$ sec

For the 3G/HSDPA network bandwidth case, Fig. 5.14 confirms that TB-BSC improves the video quality compared to TB-SVC. But we still have too much quality variations. The main reason for this is the possible capacity estimation errors due to the short-term capacity fluctuations. The throughput based method has its limitations and we resort to the buffer based method to smooth the quality.

Buffer based algorithm: Fig. 5.15 shows the requested bitrate for BB-BSC-1 and BBA-1. The buffer thresholds are $r = 20$ sec, $c_1 = 70$ sec, $c_2 = 50$ sec and the offset $\phi = 10$. BB-BSC-1 achieves a higher video quality than BBA-1. The average bitrate is 689 Kbps for BB-BSC-1 against 660 Kbps for BBA-1. There is also less quality variations for BB-BSC-1, 10 against 22 for BBA-1. BBA-1 algorithm uses r, c_1. In Fig. 5.15, $c_1 = 70$. What happens if $c_1 = 50$?, i.e., when we switch the values of c_1 and c_2. Fig. 5.16 shows the requested bitrates. The results are similar to the case $c_1 = 70$, then BB-BSC-1 still outperforms BBA-1 with 10 quality variations against 22 for BBA-1.

Fig. 5.17 and 5.18 show the requested bitrate for BBA-0 and BB-BSC-0 for different buffer thresholds. In Fig. 5.17, B_1, B_2 and B_3 are, respectively set to 5 sec, 7 sec and 50 sec. The video quality is the same for the two algorithms: 757.66 Kbps for BB-BSC-0

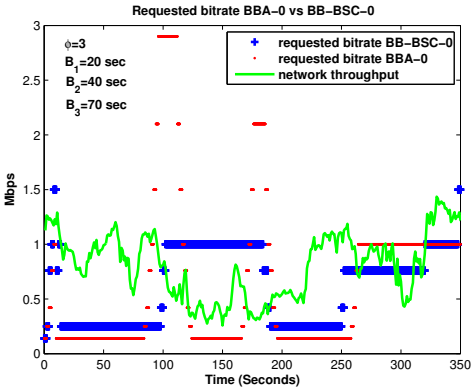

FIGURE 5.18: Requested bitrates for BBA-0 and BB-BSC-0 for $B_1 = 20$ sec, $B_2 = 40$ sec and $B_3 = 70$ sec

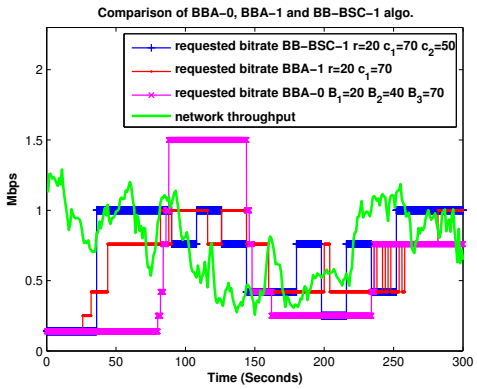

FIGURE 5.19: Comparison of BBA-0, BBA-1 and BB-BSC-1 algorithms

and 757.14 Kbps for BBA-0. The number of quality variations is 10 for BB-BSC-0 and 15 for BBA-0. In Fig. 5.18, we change the buffer thresholds to 20 sec, 40 sec and 70 sec respectively for B_1, B_2 and B_3. The video quality of BBA-0 (646 Kbps) is slightly higher than the video quality of BB-BSC-0 (624 Kbps). But BB-BSC-0 outperforms BBA-0 in terms of quality variations: 17 for BB-BSC-0 against 35 for BBA-0. We can see that the main difficulty of buffer based algorithms is the way to choose the buffer thresholds, especially when there is many buffer thresholds to set. Since BB-BSC-0 does not outperform BBA-0 in terms of video quality, we will only consider BB-BSC-1 in the remaining chapter.

Fig. 5.19 shows the requested bitrates for BB-BSC-1, BBA-1 and BBA-0. In BBA-0 algorithm, there are three buffer thresholds B_1, B_2 and B_3. To make the comparison fair, we take $B_1 = r$, $B_3 = c_1$ and make simulations to select the value of B_2 that gives the best results for BBA-0. We also remove the startup phase at the biginning of BBA-0 which quickly increases the video quality. The results of the comparison are shown in table 5.3 for 13 simulations runs. We conclude that the Backward-Shifted Coding system

	Average quality (Kbps)	Variance of the quality	Number of switchings	Number of playback interruptions
BBA-0	581	281e9	13.61	0
BBA-1	634	809e9	21.38	0
TB-BSC	**698**	950e9	58	0
BB-BSC-1	**687**	**116e9**	**10**	0

TABLE 5.3: Average of QoE metrics: Average quality, quality variability, number of switches and number of playback interruption.

(for both throughput and buffer based methods) outperforms classic DASH algorithms in terms of video quality. But in order to reduce the number of quality variations, we need to adopt a buffer based approach.

BSC offset impact on the system:
We compute the following metrics in Table 5.4: average video bitrate (in Kbps), variance of the quality, number of quality switchings, and number of playback interruptions,

for $\phi \in \{2, ..., 10\}$. The video duration is 300 sec and the segment duration is 2 sec, so the number of segment is $K = 150$. The value of ϕ begins by 2 which correspond to an offset of 1 segment. "$\phi = 1$" means an offset of 0 segment which is equivalent to DASH system. We observe that the average quality increases with ϕ and the number of quality switchings decreases.

We show in Fig.5.20 and 5.21, the requested bitrate for $\phi = 2$ and $\phi = 10$ respectively. It confirms that, when ϕ increases, the video quality increases (657 Kbps for $\phi = 2$ against 687 Kbps for $\phi = 10$) and the number of quality switchings decreases (29 for $\phi = 2$ against 10 for $\phi = 10$). One important observation relates to the first quality variation. We observe a step-like increases in the playout bitrate at the beginning of the streaming (this is the first quality variation). This is due to the dynamics of BSC video bitrate: the first $\phi - 1$ segments need to be played at the lower rate, because only base layer and possibly some enhancement layers are transmitted. Starting from block $\phi - 1$, the top layer segments with index ϕ up to $2\phi - 1$ start being received, so that the playout bitrate experiences a sharp increase.

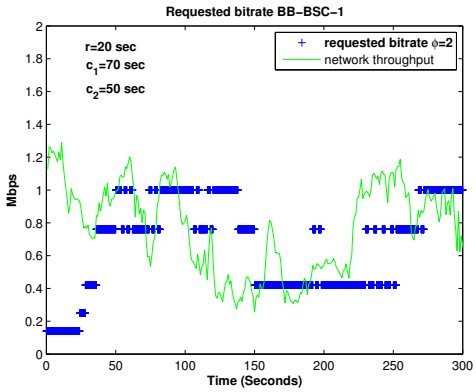

FIGURE 5.20: Requested bitrates for BB-BSC-1 for $\phi = 2$

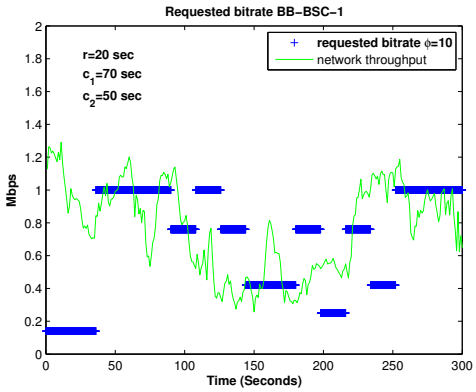

FIGURE 5.21: Requested bitrates for BB-BSC-1 for $\phi = 10$

ϕ	Average quality (Kbps)	Variance of the quality	Number of switchings	Number of playback interruptions
2	657.94	87.12e9	29.15	0
3	660.52	87.6e9	22.93	0
4	663.66	87.65e9	19.23	0
5	667.38	90.19e9	17.53	0
6	668.77	84.88e9	14.15	0
7	670.9	92.9e9	13.15	0
8	678.38	98.1e9	13	0
9	681.27	102.95e9	11.15	0
10	687.09	116.34e9	10	0
11	683.03	112.3e9	8.46	0
12	692.37	131.39e9	8.76	0

TABLE 5.4: Average QoE metrics for different values of ϕ

As mentionned before, the video file contains $K = 150$ segments. What happens if we increase ϕ up to 150. To know that, we compute the previous metrics for ϕ up

to 150 with a step size of 5. The results are shown in Fig. 5.22, 5.23, 5.24 and 5.25, respectively for the video quality, the quality switching, the variance of the quality and the number of video playback interruptions.

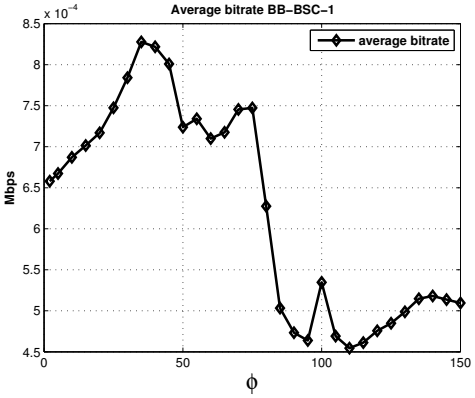

FIGURE 5.22: Average video bitrate vs the offset ϕ

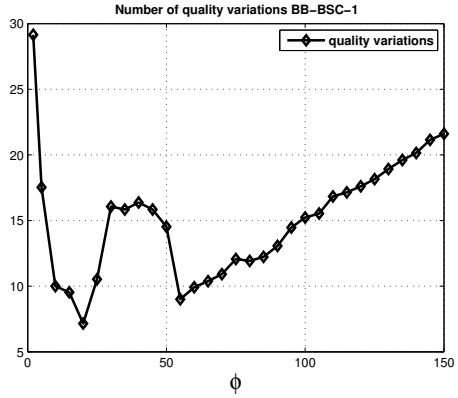

FIGURE 5.23: Average number of quality switching vs the offset ϕ

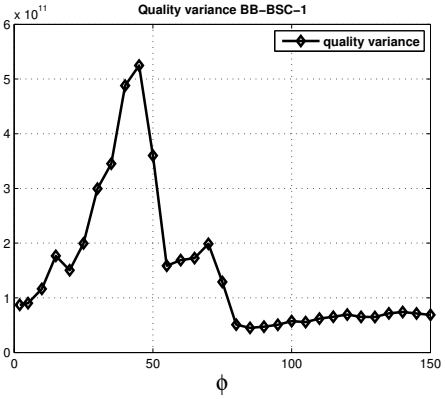

FIGURE 5.24: The quality variance vs the offset ϕ

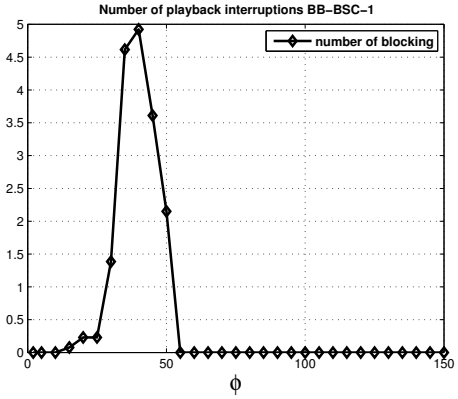

FIGURE 5.25: Average number of playback interruptions vs the offset ϕ

The first observation is that it is difficult to have a value of ϕ which optimizes all these metrics. So, we must find a tradeoff. The risk of the video playback interruption is really high for ϕ between 25 and 55. That corresponds to an offset of 50 seconds and 110 seconds duration, respectively. We must also avoid the values of ϕ such as $\phi \geq \frac{K}{2}$ (75 in this example). Indeed, for these values of ϕ, the video quality decreases and the number of quality switching increases.

In this scenario, a good tradeoff is achieved for $\phi \in \{10, ..., 25\}$. This range corresponds to $\{10, ..., 50\}$ seconds of video duration. The bounds of the range are exactly the buffer thresholds: the reservoir $r = 10$ sec and the cushion for the top layer segment $c_2 = 50$ sec. Therefore the offset ϕ depends on the buffer thresholds.

5.4 Summary

We studied the bitrate adaptation in the Backward-Shifted Coding (BSC) scheme and compared it with DASH based SVC solutions. Since BSC splits the segments into low layer segments and top layers segments and send them independently in two distinct blocks, the main challenge is how to choose the bitrates of those segments

given the network capacity which tend to highly fluctuate. Furthermore, with this time redundancy property of BSC, we are able to transmit segments and improve later their quality by sending only the appropriate number of enhancement layers.

We have proposed two bitrate adaptation algorithms, namely TB-BSC and BB-BSC, which have been designed on top of BSC. They are based on network throughput measurements and playback buffer occupancy level, respectively. We show that BSC system (HTTP adaptive video streaming system in general) may suffer from throughput estimation errors, thus, impacting the resulting QoE since we have a high number of quality variations. The limitations of the throughput based methods are overcome with the buffer based methods which set a good tradeoff between the video quality and the quality variations.

We further performed simulations compare the efficiency of BSC adaptation methods to existing DASH based SVC solutions. The results show that BSC adaptation methods achieve better video quality under same network conditions, providing a DASH-compliant solution rendering high quality video in HTTP adaptive streaming.

Since BSC is based on scalable video coding, video content caching using SVC is the interest of the next chapter.

Chapter 6

Video Caching in HAS using SVC for Wireless Networks

The Backward-Shifted Coding (BSC) system enhances the quality of experience (QoE) of video streaming users compared to existing DASH solutions. In the other hand, backhaul connection to the radio access network in turn is now becoming bottleneck. Therefore, adaptive streaming solutions may also suffer network delays resulting in the degradation of the users' quality of experience.

A solution to the ever increasing video traffic is Mobile Edge Caching (MEC). Mobile-edge caching has two additional technological motivations compared to the traditional Content Delivery Networks (CDN). First, caching videos close to eNodeBs (eNBs) and small cells (SCs) relieves congestion at the network operators' backhaul. Second, caching enables also better control of the quality of experience (QoE) of streamed videos. For example, with LTE network, end to end connections are confined within the LTE Evolved Packet System (EPS), which in turn is fully under control of the mobile network operator, taking thus full advantage of the existing EPS QoS mechanisms.

With adaptive video streaming, each video is provided in multiple copies (versions or layers). The uptake of rate-adaptive mechanisms poses the question on how to set the quality of cached copies. In this chapter, we study schemes for mobile edge caching which optimize the stored playout video rate of cached copies. We first describe the system architecture which consists on a single base station within a Macro Cell (MC) containing small cells, both access the same local edge cache. Then, we propose new Rate-Aware (RA) caching policies for layered videos and determine the static optimal caching strategy by using a Linear Programming (LP) formulation. This provides a reference bound for the performance of the proposed caching algorithms. Finally, we show through numerical simulations that the proposed caching algorithms decrease the diverted throughput from the origin server and increase the system overall hit ratio.

6.1 System description

The reference system architecture is represented in Fig. 6.1. A set $\mathcal{U} = \{1, 2, ..., U\}$ of users are requesting video files through the mobile operator network. We assume they are all associated with a tagged base station. A cache of capacity M bytes is installed at the edge of the operator network. The network of the operator is connected to the core network through a wired backhaul link. The video files are stored in the content provider video server behind the core network. Throughout this chapter, we shall adopt a bottleneck model: the capacity of the channel connecting the video server to the User Equipment (UE) is dictated by the radio link capacity.

A set $\mathcal{I} = \{1, 2, ..., I\}$ of video files are stored in the video server. In order to avoid trivial cases, we shall assume that aggregate size of the video files is higher than the cache capacity M. Also, each video can be delivered in L different quality levels. We

assume that the video files are encoded using the Scalable Video Coding (SVC). Then, there is a set $\mathcal{L}$ of layers that can be offered for each video $i \in \mathcal{I}$. Layer 1 corresponds to the base layer which renders the video with the minimum quality. In order to decode layer l, all preceding layers $l^{'} < l$ of the same video file should be available. Let $R_1, R_2, ..., R_L$ be the set of bitrates corresponding to the set of layers $\mathcal{L}$. Thus, R_l is the bitrate of the video layer l. We use the terms *layer* and *quality level* interchangeably. The playout duration of video file $i \in \mathcal{I}$ is T_i. Then the size of the video file i with the quality level or layer l is $R_l \cdot T_i$.

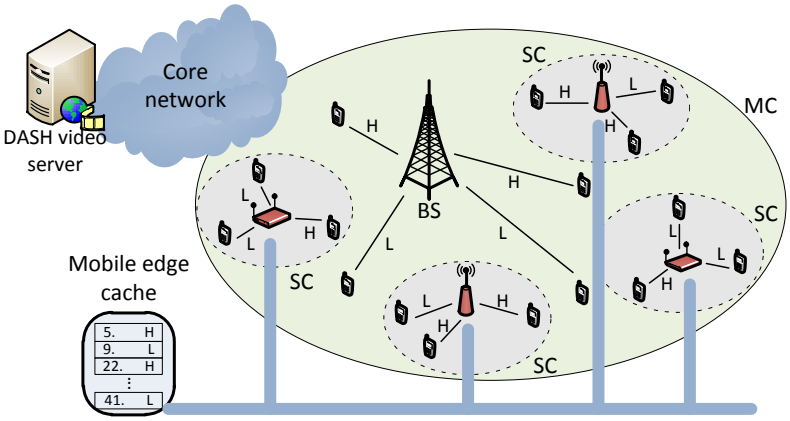

FIGURE 6.1: System Architecture: UEs are connected to the remote video server through the radio channel and a proxy of the origin video server can intercept requests using copies of the requested videos cached locally.

We let ω_i be the aggregate demand rate for video i. Then, based on the available channel quality, video applications will also associate a quality level l requested for the requested video. When the requested video is stored in the cache memory with the requested quality, it is served to the user. Otherwise, it is fetched from the origin video server.

The channel of each UE may vary and may be different for each user; we assume it takes its value in the set $\mathcal{C} = \{C_1, C_2, ..., C_L\}$. In the rest of this chapter, we assume that the radio link is the bottleneck for the system: when tagged user u faces a channel in state $C_u(t) = C_k$, for some $1 \leq k \leq L$, we assume that the playout can be performed at a specific rate R_k. We, hence, for a cached video have the set of available rates $\mathcal{R} = \{R_0, R_1, \ldots, R_L\}$, where $R_0 = 0$ means that the video is not stored in the cache.

Finally, according to SVC, if a lower quality version of the video is stored, only the missing layers have to be fetched from the video server. Hence, let us assume that a request for a tagged video is issued by a user with channel in state C_k, and that the video is stored with quality l: the throughput generated towards the server by the video request writes $[R_k - R_l]^+$, where $[b]^+ = \max\{0, b\}$. This resumes the fact that SVC cached copies stored at rate R_l require additional segments to be fetched from the remote origin server only for $R_k > R_l$.

6.2 Optimal solution using Linear Programming

In this section we compute the static optimal cache composition using a linear programming formulation by minimizing the diverted throughput from the origin server. This is equivalent to the maximization of the cache composition in order to reduce the number of sessions to the origin server. Then, we extend the problem to the lower bound case where each content stays in the cache a fraction of time.

6.2.1 Optimal static solution

We are interested in the throughput generated from the origin server: each file request generates a traffic flow, and the superposition of the flows generated by single requests offers an aggregated throughput. It represents the cost which we aim at minimizing by optimizing the cache contents. Let us denote $x_{i,l}$ the indicating variable encoding the event that content i is stored in the cache with rate l, where $\sum_{l=1}^{L} x_{i,l} = 1$. We assume that the ergodic stationary probability that a randomly sampled user has channel in state R_k is $\pi_k = \mathbb{P}\{C_u(t) = R_k\}$. Finally, let us assume that the requests of users are uniform and independent across the deployment and independent of the channel state.

In the simplest case the channel is static for every UE and n_k is the number of UEs with channel state C_k, $k = 1, \ldots, L$, we can hence write $\pi_k = n_k/U$. The Belady Algorithm [78] which is the reference optimal ideal eviction policy for non rate-aware caching policies, does not apply in our context. Hence we propose a static caching policy which requires apriori knowledge of popularity: we shall use it in order to compare the performance of dynamic policies described in the rest of the chapter.

The expected throughput generated to the origin server by using SVC-based RA caching policies writes

$$
\begin{aligned}
\mathrm{E}[\tau_i] &= \omega_i T_i \sum_{k=1}^{L} \left[R_k - \sum_{l=0}^{L} R_l \, x_{i,l} \right]^+ \pi_k \\
&= \omega_i T_i \sum_{k=1}^{L} \Big(R_k - \sum_{l=0}^{k-1} R_l x_{i,l} - \sum_{l=k}^{L} R_l x_{i,l} \Big) \pi_k \\
&= \omega_i T_i \left[\sum_{k=1}^{L} \sum_{l=0}^{k} \Big(\frac{R_k}{k+1} - R_l \cdot x_{i,l} \Big) \pi_k - \sum_{k=1}^{L} R_k \pi_k \sum_{l=k}^{L} x_{i,l} \right] \\
&= \omega_i T_i \left[\sum_{l=1}^{L} \sum_{k=l}^{L} \Big(\frac{R_k}{k+1} - R_l \cdot x_{i,l} \Big) \pi_k + \sum_{k=1}^{L} \frac{R_k}{k+1} \pi_k \right. \\
&\quad \left. - \sum_{l=1}^{L} \sum_{k=1}^{l} R_k \pi_k x_{i,l} \right] \\
&= \omega_i T_i \Big(\sum_{l=1}^{L} \sum_{k=l}^{L} \frac{R_k}{k+1} \pi_k + \sum_{k=1}^{L} \frac{R_k}{k+1} \pi_k \Big) \\
&\quad - \omega_i T_i \sum_{l=1}^{L} \Big(R_l \sum_{k=l}^{L} \pi_k + \sum_{k=1}^{l-1} R_k \pi_k \Big) x_{i,l}
\end{aligned}
$$

We observe that the first term does not depend on the caching policy. Hence, our goal is to maximize the objective function

$$\Lambda = \sum_{i=1}^{I} \Lambda_i = \sum_{i=1}^{I} \omega_i T_i \sum_{l=1}^{L} E_l \, R_l \, x_{i,l} \tag{6.1}$$

where

$$E_l = R_l \sum_{k=l}^{L} \pi_k + \sum_{k=1}^{l-1} R_k \pi_k$$

Equation (6.1) represents the expected *cache throughput* Λ, i.e., the expected throughput that we are able to divert from the flow directed to the remote server and serve instead by using the local cache.

The solution is given in the simplest case by static caching policy $x_{i,l}$, for which the problem can be formulated as a Multiple Choice Knapsack Problem (MCKP):

Problem 1. *Determine the static content allocation strategy that maximizes*

$$\textit{maximize} \quad \sum_{i=1}^{I} \omega_i T_i \sum_{l=0}^{L} E_l \, R_l \, x_{il} \tag{6.2}$$

$$\textit{subject to} \quad \sum_{i=1}^{I} \sum_{l=1}^{L} T_i \, R_l \, x_{il} \leq M, \tag{6.3}$$

$$\sum_{l=0}^{L} x_{il} = 1, \quad i = 1, 2, ..., I, \tag{6.4}$$

$$x_{il} \in \{0, 1\}, i = 1, 2, ..., I, \; l = 1, 2, ..., L \tag{6.5}$$

where M is the cache size; now decision variable x_{il} means content i is stored with quality l. The condition (6.4) means that only one quality of content i is cached at a time.

6.2.2 Lower bound solution

We observe that (6.5) can be relaxed, i.e., we can let $x_{il} \in [0, 1]$ a continuous decision variable. In particular, this suggests that content i can be stored with different playout rate qualities for a certain fraction of the time. For instance, if $\mathcal{R} = \{0, R_1, R_2\}$, and $x_{i,0} = 0.1$, $x_{i,1} = 0.3$, and $x_{i,2} = 0.6$, it means that content i should be stored in the cache with the higher quality for 60% of the time, whereas for the 30% of the time it should be stored with the lower quality, and for the remaining fraction of the time it should not be stored at all. If we solve Problem. 1 with relaxed decision variables, we obtain indeed an optimal solution, i.e., possibly better that the optimal static policy. However, it may not be possible to implement it: in fact, we have no guarantee that given the optimal solution, we can arrange the contents in the cache such in a way to satisfy at each point in time the cache occupancy constraint (6.3). However, the solution so derived represents a lower bound for the expected cache throughput.

6.3 Our proposed caching algorithms

In this section, we first adapt existing caching policies to the case of scalable video coding. We derive new rate-aware caching policies. Then, we propose a channel quality matching algorithm which follows the distribution of the channel quality.

6.3.1 Quality-aware algorithms using existing caching policies

We show how existing caching policies such as LRU and LFU can be adapted to the case of scalable coding videos. We consider LRU and LFU for simplicity's sake but the same adaptation can be applied to other existing caching policies such as FIFO, LRFU, k-LRU.

Classical LRU and LFU policies, when used in single version or multiple versions videos, work in the following way: under the LRU policy, the cache stores each video from incoming requests at the top of the cache. In case the video version is already in the cache it is simply moved at the top of the cache. If not, it is retrieved first from the origin server. When the cache is full, the content from the bottom of the cache is evicted to make space for the new one. In the same way, with LFU policy, the most popular videos are kept in the cache and an incoming video request enters in the cache only if it is most requested than the less popular content in the cache. And, such less popular content is evicted from the cache.

Observe that existing caching policies can be used with multiple versions videos because each version can be regarded as a different content. In this case, we can have several versions of the same video in the cache. The purpose of [89], in particular, was to find which versions store in the cache in order to optimize the cache allocation strategy. In the remainder of the chapter, we use LRU and LFU policies for video versions, i.e., when the videos are encoded using AVC. In scalable coding videos, conversely, each layer corresponds to a given video quality. But, a layer cannot be decoded alone to render the corresponding quality. In fact, in order to decode the layer l, all preceding layers $l' \leq l$ of the same video file should be available. Then, the layers are not independent and we cannot use the existing caching policies alone. But these policies will be the basis of our proposed caching policies.

Which quality to store?: Most Recently Quality vs Most Frequently Quality

We build new caching policies for layered videos. The existing caching strategies are the basis of the new ones because regardless the requested quality, we can still use policies such as LRU or LFU to store the contents in the cache or evict them from the cache. Then, the fundamental question is with quality to store in the cache. A natural answer to this question is obviously the quality which is requested. Since the users requesting the same video content do not experience the same channel conditions (they are more or less closed to the base station), we consider two caching methods: the Most Recently Quality (MRQ) and the Most Frequently Quality (MFQ).

MRQ: In the Most Recently Quality, we store the requested video with the requested quality. Then each time the video is requested with a different quality, we updated the quality in the cache by reducing or increasing the quality. We delete enhancement layers to reduce the quality and to increase the quality, we retrieve the necessary enhancement layers from the server.

MFQ: In the Most Frequently Quality, we store the quality which is the most requested for each video. We denote by Q_{req} and Q_{freq} the requested quality by the user and the most frequently requested quality for that video. If $Q_{req} = Q_{freq}$, MFQ is equivalent to MRQ. But if these two qualities are different, MFQ caching strategy acts differently from MRQ. For instance, if $Q_{req} > Q_{freq}$, MFQ will retrieve the missing part from the server.

Recall that the proposed caching methods work on the quality and not on the videos since the classical caching algorithms can still be used to store and evict the

contents. For example when using LRU policy for the videos, we refer to LRUMRQ and LRUMFQ as the caching methods for the videos and the qualities together. In the same way, LFUMRQ and LFUMFQ will be the caching methods when using LFU for the videos.

How to evict contents?: Deleting vs Trimming

Herein, we exploit the benefit of the scalability of layered videos to evict the contents from the cache. Since each video in the cache is composed of layers, we can trim the content by reducing its quality instead of automatically discarding the entire video file (the base layer and all its enhancement layers in the cache). Each method has its advantages and drawbacks. When deleting the entire video file, we allow other videos to be stored with high quality. On the other hand, when reducing the quality, we increase the hit ratio of the system. We will compare the performance of these two eviction methods.

6.3.2 Channel quality matching (CQM) caching algorithm

So far, in the proposed algorithms, the videos are correlated with the qualities. But in reality, users request the videos regardless their network channel states. Then, the videos popularity may not be correlated or weakly correlated to the channel qualities. By understanding this property, we propose an online channel quality matching (CQM) algorithm in order to follow the network channel distribution in the cache. Let V and v_l be, respectively, the total number of videos in the cache and the number of videos with quality l in the cache. We have $V = \sum_{l=1}^{L} v_l$. We define s_l to be the size in bytes of any video with quality l (assuming that videos have the same duration), then

$$\sum_{l=1}^{L} v_l s_l = M \tag{6.6}$$

where M is the cache size. Since the goal of CQM algorithm is to match the network channel distribution, we have

$$v_l = V.\pi_l \tag{6.7}$$

where π_l, defined in Section 6.2.1, is the probability that the channel is in state R_l. We learn π_l online by recording the number of requests for each quality regardless the videos. Using Eq. 6.6 and 6.7, we derive V:

$$V = \frac{M}{\sum_{l=1}^{L} s_l \pi_l} \tag{6.8}$$

Finally, we get v_l by using Eq. 6.8 in Eq. 6.7. CQM algorithm gives the number of videos of each quality in the cache but it does not specify which videos. Therefore, we keep in the cache the most popular videos with high qualities in LFU manner.

The channel quality matching caching algorithm is illustrated in Fig. 6.2. First, the videos are ranked by popularity by recording the number of requests for each video. This ranking must be updated for each new request. Then, the channel distribution is used to compute the number of videos for each quality by using the above equations. Finally, we cache the most popular videos following this distribution. For instance, in Fig. 6.2, we have, in the cache, 4 videos with quality $720p$, 5 videos with quality $480p$ and 2 videos with quality $360p$.

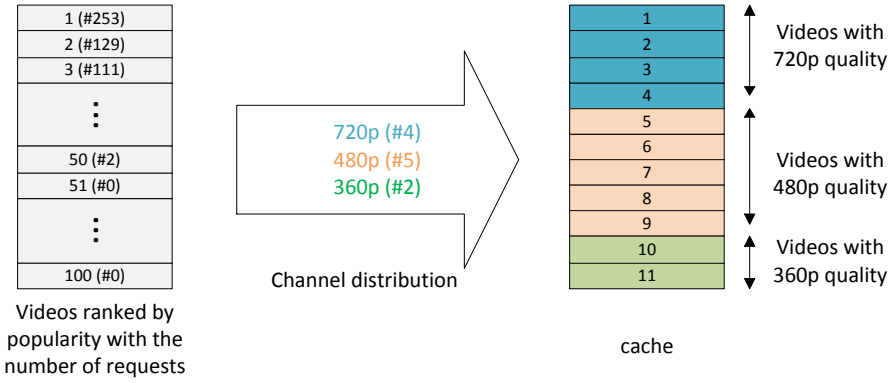

FIGURE 6.2: Channel Quality Matching caching algorithm

6.4 Simulations and numerical results

6.4.1 Simulation setup

We build a discrete event simulator according to the system architecture of Fig. 6.1: A single base station with mobile users within the cell range asking for video contents. The set of video contents are available at the origin server with different quality levels/ representations/ bitrates (versions and layers). We consider 100 videos, each video has five (5) quality levels, i.e., {250; 400; 750; 1000; 1200} Kbps and 300 seconds duration.

The video requests arrive to the base station according to the well-known Independent Reference Model (IRM). We assume a Zipf's distribution as the video content popularity law. The Zipf's distribution states that the probability to request the i-th most popular video content is proportional to $1/i^{\alpha}$, where α is the Zipf's distribution exponent. Unless specified, we will always consider a catalog (number of requests) size of 10^7, and a Zipf's distribution exponent $\alpha = 0.5$. We use a uniform distribution for the requested qualities.

Layered videos encoding process generates encoding overhead compared to single layer codec. Typically, for the same quality level, the total rate for the base and enhancement layers combined is larger than the rate of the same quality version. Let s_{il} denote the size of layer l of video file i and s_{iq}^{v} denote the size of the version v of video file i with quality q. Then, the size of the layered video of quality q is given by [92]:

$$\sum_{l=1}^{q} s_{il} = s_{iq}^{v}(1 + OV_i^q), \ \forall \ q \ \in \{1, 2, ..., L\} \tag{6.9}$$

where $OV_i^q \geq 0$ is the encoding overhead for the quality level q of video file i. Whereas, in the literature, people usually assumes 10% encoding overhead per enhancement layer compared to single-layer AVC, authors in [106, 107] recommend the following: they keep the original bitrate for the base layer, i.e.,$OV_i^1 = 0$. They increase the bitrate for the first enhancement layer by 10%, for the second layer by 20% and for the third layer for 30%. Since we have five quality levels, we use the vector **OV**= [0 0.1 0.2 0.3 0.4] for the encoding overhead of SVC.

6.4.2 Numerical results

Fetched content from the origin server: We first compare the two caching policies: the Most Recently Quality (MRQ) and the Most Frequently Quality (MFQ). Recall that MRQ stores the content with the last requested quality whereas MFQ stores the content with the quality which is the most requested. We use LRU and LFU policies for the eviction. Fig. 6.3 compares MRQ and MFQ for LRU and LFU policies in terms of fetched content (in Gbytes) from the origin server. MRQ is better for LRU while MFQ is better for LFU. Then, the quality to store depends on how the content is stored and the eviction policy. We observe also that LRUMRQ is better than LFUMFQ when the cache size is small. This is due to the fact that with MFQ policy, the requested quality may be higher than the quality to store in the cache. In that case, the content should be retrieve from the server to satisfy the user request. In the remaining Figures, we only plot LRUMRQ policy for the comparison with the other caching policies for readability issue.

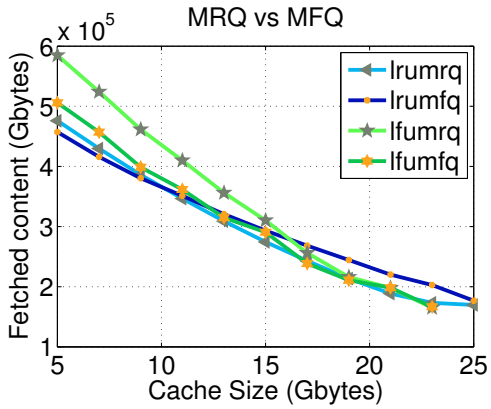

FIGURE 6.3: Most Recently Quality vs Most Frequently Quality

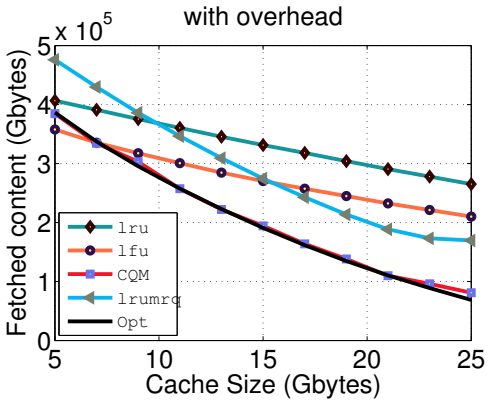

FIGURE 6.4: Comparison of caching policies: fetched content

In Fig. 6.4, we plot the fetched content in Gbytes for all the caching policies including the optimal static solution given by the Linear Programming. LRU and LFU policies are the standard caching policies used for video versions. We first observe that our caching policies outperform the standard caching policies except when the cache size is small. For example, CQM algorithm does not perform better than LFU when the cache size is 7GB (13% of the total videos size). This may due to the encoding overhead of scalable video coding. We plot the fetched content without encoding overhead in Fig. 6.5. As expected, our caching policies outperform standard caching policies for small cache regime. The static optimal policy stores the popular videos with high quality and decreases the quality for the less popular irrespective to the cache size. The closest policy to the static optimal caching is CQM algorithm. This is because, CQM also stores the popular videos with the highest quality among the requested qualities. For instance, if the channel state quality is unique, CQM is equal to the optimal solution since it stores only that quality in the cache. When the number of videos for each quality in the cache are the same, CQM works like *Partitioned* algorithm in [91] which also stores the same number of videos for each quality representation (it uses versions). But *Partitioned* will be no longer efficient when the requests are not uniformly distributed on the qualities. On the other hand, CQM will be still close to the optimal solution since it follows the channel distribution.

Hit ratio: The hit ratio of a request is the ratio between the quality that the request

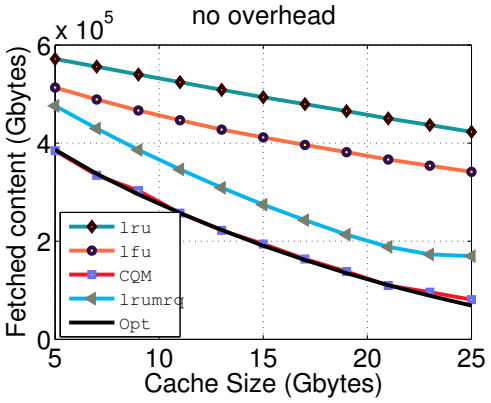

FIGURE 6.5: Comparison of caching policies assuming no overhead in SVC encoding process

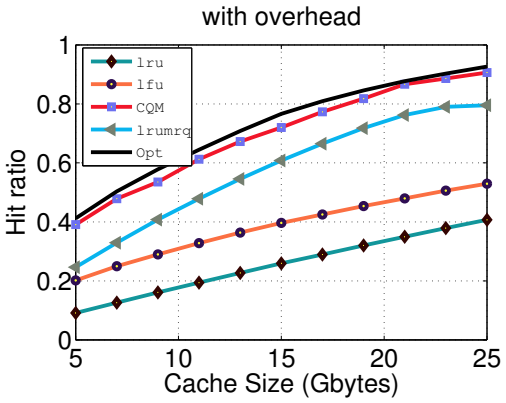

FIGURE 6.6: Comparison of caching policies: hit ratio

finds in the cache and the requested quality. When using versions, the hit ratio of a request is 0 if the requested quality is not in the cache and 1 otherwise. But, when using layers, the request can find a lower quality than the one asked. In that case, the base station has to retrieve from the server only the missing part. Then the hit ratio is between 0 and 1. Fig. 6.6 shows the hit ratio for the caching policies. Our rate-aware caching policies outperform LRU and LFU. It confirms that CQM is close to the static optimal solution.

Fig. 6.7 shows the hit ratio for LRUMRQ caching policy when deleting and trimming contents from the cache. Deleting contents is the standard eviction method: given the eviction policy, we discard the content from the cache. For example, LRU discards the last content in the cache while LFU discards the less popular content in the cache. One more flexibility that gives SVC is trimming contents: reduce the quality of the content by discarding the enhancement layers. In Fig. 6.7, we reduce the quality of the last content until the base layer. We repeat this action for the next contents until there is enough space to store the new content. We increase the hit ratio by trimming contents.

Cache occupancy and videos' popularity: Fig. 6.8 shows the proportion of

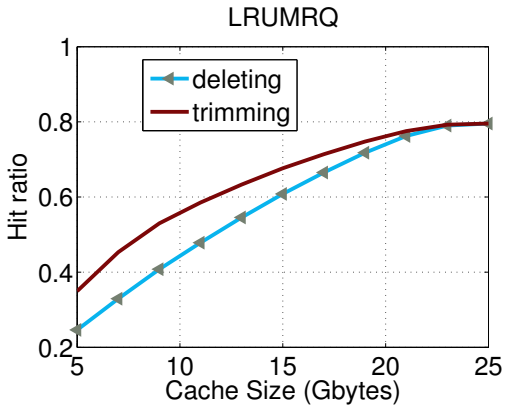

FIGURE 6.7: Eviction methods: deleting vs trimming

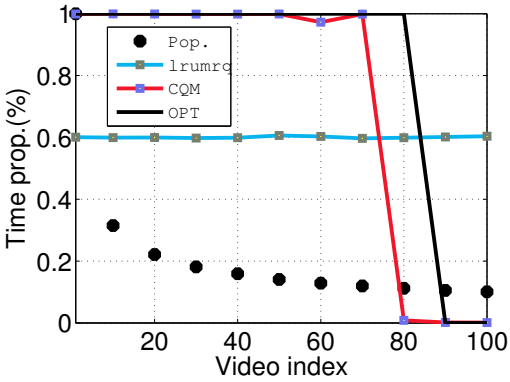

FIGURE 6.8: Cache occupancy vs video popularities

time spent by video i with the quality of 750Kbps for a cache size of 25Gbytes, where $i = \{1, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$. Our first observation is that good

caching policies always keep most popular videos in the cache. LRUMRQ does not care about the popularity since the curve is flat. This reduces its performance. The optimal solution and CQM algorithm always put the most popular videos in the cache irrespective to the channel distribution and the cache size, thus, reducing the diverted throughput from the origin server.

6.5 Summary

Caching is a topic which regains attention in recent years, especially for video contents. Indeed, network operators are anxious about their backbone network since the backhaul links are becoming the bottleneck of the transmission. This is mainly due to the huge amount of video data that is delivered to the end users. Unfortunately, existing contents caching policies are not adapted to the context of HTTP adaptive streaming because the content is not anymore a unique object. In HAS, the same video is provided in multiple representations for multiple qualities.

We introduced a set of new rate-aware caching policies. They account for both video popularity and for the site-dependent link quality footprint. Such policies optimize the playout rate for videos stored while maximizing the cache throughput. By doing so, they also reduce the experienced video delay and divert most of the throughput directed to remote video servers, which is responsible for a significant fraction of the mobile network traffic. These new rate-aware caching policies include modified versions of existing caching policies such as LRU, LFU, and caching policies that account for the users channel distribution. In the former caching policies, we use existing caching policies to store and evict the content from the cache. For the quality of the content to store, we propose two methods called the Most Recently Quality (MRQ) and the Most Frequently Quality (MFQ). In the latter caching policies, we take into account the channel distribution. Thus, the number of contents and qualities in the cache are governed by the channel. We hence reproduce the channel state into the cache. We also define an optimization problem using the linear programming in order to minimize the diverted throughput from the origin server. This static optimal solution is used to compare the efficiency of our proposed rate-aware caching policies.

Numerical results indicate that such schemes outperform rate-agnostic caching policies. Also, we have shown that large performance gains are obtained by adapting customary algorithms such LRU or LFU with ad hoc policies which decide either contents' eviction or contents' trimming based on the available video representations stored in the cache. Further enhancement require to match the stored quality to the users' link quality distribution.

Chapter 7

Conclusions and Future Works

Designing and improving video streaming solutions become crucial as video traffic dominates the Internet. This traffic will reach 75% of the total Internet traffic by 2020 according to Cisco. As a consequence, it becomes challenging to satisfy video consumers. Further, backhaul links are also becoming the bottleneck of the transmission at network side. In this thesis, we propose two solutions to improve video streaming delivery on mobile networks: the Backward-Shifted Coding (BSC), a novel video transmission scheme using the Scalable Video Coding (SVC) and new rate-aware caching algorithms that accounts for the diversity of the channels connecting the video server to the users.

In **Chapter 3**, we introduced the Backward-Shifted Coding scheme. It uses the capabilities and the flexibility of SVC codec to introduce an offset between the transmission of the video segments. This offset ϕ is an important parameter in BSC system. The scalable video coding is a multi-layer codec that encodes the video segments in multiple layers: one base layer and multiple enhancement layers. The base layer renders the video with the minimum quality and the enhancement layers increase the video quality. BSC sends a complete segment (low layer segment $k + \phi - 1$) with the enhancement layers of another segment (top layer segment k). We showed that BSC is fully client driven since it does not require any modifications on the media server side.

In **Chapter 4**, we adopted a QoE-based performance analysis of BSC system. We model the playback buffer as an M/M/1 queue. Firstly, we compute close form expressions of the buffer starvation probability using the Ballot theorem and derive the probability generating functions of the starvation events, i.e., the number of buffer starvations during the video session. Further, we compute the startup delay using the prefetching process at the video session initiation. The results showed that BSC scheme greatly decreases the probability of buffer starvation compared to classic SVC codec since it is possible to switch to the low layer segments when the top layer segments are missing. We also provide the range of values of ϕ which minimize the buffer starvation probability. Secondly, we model the number of video frames of each quality in the buffer and use quasi-stationary distribution of birth-death Markov processes to compute the average video quality in BSC system. We finally use a QoE cost function including the computed metrics to analyse the performance of BSC system. The results showed that BSC can improve the video quality while maintaining the same risk of video playback interruption compared to classic SVC codec.

In **Chapter 5**, we proposed quality adaptation algorithms in order to fully exploit the benefits of BSC and compared them to existing algorithms in DASH. Compared to existing DASH systems where the adaptation is done on single segment basis, our proposed adaptation algorithms have to select the bitrates for the low layer segment and

the top layer segment at the same time. By doing so, we can decide the quality of the segment first and adjust this quality later with the top layer segment. This capability of BSC introduces more flexibility and robustness in the selection of the segment quality. The results showed that we increase the video quality by almost 19% when we use the network estimated throughput as feedback in the adaptation algorithms. Further, the quality variability smoothing method that we used in the buffer-based algorithms improves the quality switching up to 36%.

In **Chapter 6**, we proposed new rate-aware caching policies in HTTP adaptive streaming context. Existing policies fail to optimize caches composition since in HAS systems, the same video is provided in multiple representations. We consider a mobile network where the requested video quality of a user is governed by its network channel. Further, the videos are encoded using the scalable video coding. We propose several algorithms that optimize the cache composition. The first group of algorithms uses existing caching policies, i.e., LRU, LFU, etc. The idea is to use these algorithms for the content (regardless the quality) and use other methods for the quality to store in the cache. We propose the Most Recently Quality (MRQ) and the Most Frequently Quality (MFQ). The results showed that MRQ and MFQ policies increase the system performance compared to existing caching policies such as LRU and LFU. Our second proposed algorithm (CQM) exploits the distribution of the users channel rates. The idea is to match the channel distribution with the cache composition while maintaining the most popular contents in the cache. We determine the static optimal caching strategy by using a Linear Programming (LP) formulation in order to compare it with our proposed algorithms. The results showed that CQM algorithm performs as well as the optimal static solution. We also showed that we increase the system performance by trimming the contents in the cache.

Future works

Quality of Experience: An interesting perspective about the quality of experience is the development of a general standard QoE framework because there is a lack of consensus on Internet video QoE. This idea is challenging since QoE is a subjective measure. We propose the development of a general function which will be a mapping function between QoE and QoS. Further, to account for the subjective nature of the quality of experience we will incorporate the user preferences on this QoS mapping function. Hence, it will be easy to compute the QoE since the QoS is expressed by the network parameters and the users preferences can be set on video players and obtain by recording clients historic and logs.

HTTP Adaptive Streaming: So far, we study BSC system and DASH system in general for a single user. An interesting perspective should be the behavior of BSC system when multiple users compete for the network bandwidth. The main issue in such competition scheme is the stability of the bitrate. In fact, some users may overestimate or underestimate the throughput when the other users pause the segments downloading to control the level of the playback buffer. We could see if our bitrate stability function still works.

Otherwise, a scheduling technique at the base station should be necessary. That introduces the problem of resource allocation in HTTP adaptive streaming. The joint optimization of resource allocation and video QoE should be an interesting problem in BSC.

There is a lack of models in the literature for the bitrate adaptation problem. In our future works, we will propose to model the problem of bitrate adaptation using Markovian models and deep learning approach in order to derive some interesting and useful parameters. In Backward-Shifted Coding system, there are some important parameters such as the offset ϕ or the buffer thresholds in the case of the buffer based algorithms. We showed in our results that the video quality may depend on these buffer threshols. Further, it is difficult to set these thresholds in runtime systems without any test before. With bitrate adaptation models, it may be easy to choose these buffer thresholds. We also showed that the offset ϕ is difficult to choose because a single value does not optimize all QoE metrics simultaneously. A good bitrate adaptation model may facilitate these limitations.

Another future work in BSC could be the comparison of BSC with HAS systems which assume knowing the channel state in advance. Indeed, BSC does not assume knowledge of the future channel state but it does an interesting thing which is the possibility to adjust the segment quality in the future since it does the adaptation twice. We believe this capability of BSC corrects the throughput estimation errors, hence it could compete with HAS systems which assume knowing the future, at least for a few seconds.

Video caching: In our HAS system, we consider the average user channel rate that is used to request the appropriate video quality in the cache (or directly in the distant server). But actually, the quality may change per segment basis. So one of our future works is to account for the channel rate variability in the caching decision. Knowing that the channel rate is available after the download of each segment, which quality should we select for the video to be store in the cache? Should we take all these informations in the caching decision because there may be memory and speed constraints.

In our future works we will also account for the time varying popularity of the contents. Indeed, in today complex systems, the popularities of the videos are not only static. A typical example is about TV series. When a new season appears, the popularity increases quickly and decreases after some time. But before the next season appears, the popularity of the previous season increases again because people want to watch it again in order to remenber the story.

Further, we will explore the performance of our proposed algorithms for interconnected caches, e.g., many cache servers are deployed in the network. In this case, we have to decide not only the caching policies (number of copies to cache) but also where to cache the contents and how to replicate them. In the case of BSC, we may store the layers (base layer and enhancement layers) of a single video in different caches.

Another future work is about developing mathematical models for optimal caching using for example dynamic programming. So, we can derive some useful metrics such as the system hit probability.

So far in the literature, the optimal caching soluton is derived by minimizing or maximizing a single QoE or QoS metric such as the video quality, the ratio between the requested bitrate and the rendered bitrate, the delay from the origin server, the throughput diverted from the origin server or the network operator cost. In our future works, we propose to derive the optimal caching solution in order to maximize the users quality of experience including the video quality, the quality variations, the initial startup delay and mostly the risk of video playback interruptions.

Bibliography

[1] Cisco Visual Networking Index. "Cisco visual networking index: Global mobile data traffic forecast update, 2015-2020 white paper". In: *Cisco Systems* ().

[2] Erik Dahlman et al. *3G evolution: HSPA and LTE for mobile broadband*. Academic press, 2010.

[3] Zakaria Ye, Tania Jiménez, and Rachid El-Azouzi. "Video streaming analysis in Vienna LTE system level simulator". In: *Proceedings of the 8th International Conference on Simulation Tools and Techniques*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). 2015, pp. 47–54.

[4] Josep Colom Ikuno, Martin Wrulich, and Markus Rupp. "System level simulation of LTE networks". In: *Vehicular Technology Conference (VTC 2010-Spring), 2010 IEEE 71st*. IEEE. 2010, pp. 1–5.

[5] ITURM ITU. "2134, Requirements related to technical performance for IMT-Advanced radio interface (s)". In: *International Telecommunications Union* (2008).

[6] Henning Schulzrinne et al. "Real-time transport protocol". In: *RFC1899* (2003).

[7] European Telecomunications Standard Institute (ETSI) (2010). *Universal Mobile Telecommunication System (UMTS); LTE; Transparent End-to-End Packet Switched Streaming Service (PSS): Progressive download and dynamic adaptive streaming over HTTP (3GP-DASH)*, Sophia-Antipolis Cedex, France, 3GPP TS 26.247 Version 1.0.0 Release 10.

[8] ISO/IEC. "Information Technology - Dynamic adaptive streaming over HTTP (DASH) - Part 1: Media Presentation Description and Segment Formats". In: 2012.

[9] Jirka Klaue, Berthold Rathke, and Adam Wolisz. "Evalvid–A framework for video transmission and quality evaluation". In: *International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*. Springer. 2003, pp. 255–272.

[10] International Standards Organisation/International Electrotechnical Commission (ISO/IEC). "Coding of audio-visual objects-Part 10: Advanced video coding". In: 2012.

[11] Anthony Vetro, Thomas Wiegand, and Gary J Sullivan. "Overview of the stereo and multiview video coding extensions of the H. 264/MPEG-4 AVC standard". In: *Proceedings of the IEEE* 99.4 (2011), pp. 626–642.

[12] J-R Ohm et al. "Comparison of the coding efficiency of video coding standards—including high efficiency video coding (HEVC)". In: *Circuits and Systems for Video Technology, IEEE Transactions on* 22.12 (2012), pp. 1669–1684.

[13] Heiko Schwarz, Detlev Marpe, and Thomas Wiegand. "Overview of the scalable video coding extension of the H. 264/AVC standard". In: *IEEE Transactions on circuits and systems for video technology* 17.9 (2007), pp. 1103–1120.

[14] Gary J Sullivan et al. "Overview of the high efficiency video coding (HEVC) standard". In: *IEEE Transactions on circuits and systems for video technology* 22.12 (2012), pp. 1649–1668.

[15] Yohaan Yoon et al. "Performance analysis of h. 264/avc, h. 264/svc, and vp8 over ieee 802.11 wireless networks". In: *Computers and Communications (ISCC), 2012 IEEE Symposium on*. IEEE. 2012, pp. 000151–000156.

[16] Oliver Hohlfeld et al. "A QoE perspective on sizing network buffers". In: *Proceedings of the 2014 Conference on Internet Measurement Conference*. ACM. 2014, pp. 333–346.

[17] Nokia. "Quality of Experience of mobile services: Can it be measured and improved?" In: *Nokia white paper* (2005).

[18] Patrick Le Callet, Sebastian Möller, Andrew Perkis, et al. "Qualinet white paper on definitions of quality of experience". In: *European Network on Quality of Experience in Multimedia Systems and Services (COST Action IC 1003)* (2012).

[19] Markus Fiedler, Sebastian Möller, and Peter Reichl. "Quality of experience: From user perception to instrumental metrics (Dagstuhl Seminar 12181)". In: *Dagstuhl Reports* 2.5 (2012).

[20] Markus Fiedler, Tobias Hossfeld, and Phuoc Tran-Gia. "A generic quantitative relationship between quality of experience and quality of service". In: *IEEE Network* 24.2 (2010), pp. 36–41.

[21] Hyun Jong Kim and Seong Gon Choi. "QoE assessment model for multimedia streaming services using QoS parameters". In: *Multimedia Tools and Applications* 72.3 (2014), pp. 2163–2175.

[22] ITU-T Recommendation P.800. "Methods for subjective determination of transmission quality". In: (1996).

[23] Athula Balachandran et al. "Developing a predictive model of quality of experience for internet video". In: *ACM SIGCOMM Computer Communication Review*. Vol. 43. 4. ACM. 2013, pp. 339–350.

[24] Athula Balachandran et al. "A quest for an internet video quality-of-experience metric". In: *Proceedings of the 11th ACM workshop on hot topics in networks*. ACM. 2012, pp. 97–102.

[25] Florin Dobrian et al. "Understanding the impact of video quality on user engagement". In: *ACM SIGCOMM Computer Communication Review* 41.4 (2011), pp. 362–373.

[26] S Shunmuga Krishnan and Ramesh K Sitaraman. "Video stream quality impacts viewer behavior: inferring causality using quasi-experimental designs". In: *Networking, IEEE/ACM Transactions on* 21.6 (2013), pp. 2001–2014.

[27] François Baccelli and William A Massey. "A sample path analysis of the M/M/1 queue". In: *Journal of Applied Probability* (1989), pp. 418–422.

[28] Søren Asmussen. *Ruin probabilities*. World scientific Publishing Company, 2000.

[29] Ali ParandehGheibi et al. "Avoiding interruptions—A QoE reliability function for streaming media applications". In: *IEEE Journal on Selected Areas in Communications* 29.5 (2011), pp. 1064–1074.

[30] Guanfeng Liang and Ben Liang. "Effect of delay and buffering on jitter-free streaming over random VBR channels". In: *IEEE transactions on multimedia* 10.6 (2008), pp. 1128–1141.

[31] Thomas Stockhammer, Hrvoje Jenkac, and Gabriel Kuhn. "Streaming video over variable bit-rate wireless channels". In: *IEEE Transactions on Multimedia* 6.2 (2004), pp. 268–277.

[32] Yuedong Xu et al. "Impact of flow-level dynamics on QoE of video streaming in wireless networks". In: *INFOCOM, 2013 Proceedings IEEE*. IEEE. 2013, pp. 2715–2723.

[33] Yuedong Xu, Yipeng Zhou, and Dah-Ming Chiu. "Analytical QoE models for bit-rate switching in dynamic adaptive streaming systems". In: *Mobile Computing, IEEE Transactions on* 13.12 (2014), pp. 2734–2748.

[34] Zakaria Ye, EL-Azouzi Rachid, and Tania Jimenez. "Analysis and modelling Quality of Experience of video streaming under time-varying bandwidth". In: *Wireless and Mobile Networking Conference (WMNC), 2016 9th IFIP*. IEEE. 2016, pp. 145–152.

[35] Qiang Ren and Hisashi Kobayashi. "Transient solutions for the buffer behavior in statistical multiplexing". In: *Performance evaluation* 23.1 (1995), pp. 65–87.

[36] A Narayanan and VG Kulkarni. "First passage times in fluid models with an application to two priority fluid systems". In: *Computer Performance and Dependability Symposium, 1996., Proceedings of IEEE International*. IEEE. 1996, pp. 166–175.

[37] Joseph Abate, Gagan L Choudhury, and Ward Whitt. "An introduction to numerical transform inversion and its application to probability models". In: *Computational probability*. Springer, 2000, pp. 257–323.

[38] Joseph Abate and Ward Whitt. "A unified framework for numerically inverting Laplace transforms". In: *INFORMS Journal on Computing* 18.4 (2006), pp. 408–421.

[39] Tom H Luan, Lin X Cai, and Xuemin Shen. "Impact of Network Dynamics on User's Video Quality: Analytical Framework and QoS Provision". In: *Multimedia, IEEE Transactions on* 12.1 (2010), pp. 64–78.

[40] William Feller. *An introduction to probability theory and its applications: volume I*. Vol. 3. John Wiley & Sons London-New York-Sydney-Toronto, 1968.

[41] JiaFu He and Khosrow Sohraby. "A new analysis framework for discrete time queueing systems with general stochastic sources". In: *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*. Vol. 2. IEEE. 2001, pp. 1075–1084.

[42] Pierre Humblet, Amit Bhargava, and Michael G Hluchyj. "Ballot theorems applied to the transient analysis of nD/D/1 queues". In: *IEEE/ACM Transactions on Networking (TON)* 1.1 (1993), pp. 81–95.

[43] Omer Gurewitz, Moshe Sidi, and S Cidon. "The ballot theorem strikes again: Packet loss process distribution". In: *IEEE Transactions on Information Theory* 46.7 (2000), pp. 2588–2595.

[44] Yuedong Xu et al. "QoE analysis of media streaming in wireless data networks". In: *NETWORKING 2012*. Springer, 2012, pp. 343–354.

[45] Yuedong Xu et al. "Analysis of Buffer Starvation with Application to Objective QoE Optimization of Streaming Services". In: *Multimedia, IEEE Transactions on* 16.3 (2014), pp. 813–827.

[46] Move Networks. *Move Networks 2010*. [Online]. Available: http://www.movenetworkshd.com.

[47] David F Brueck and Mark B Hurst. *Apparatus, system, and method for multi-bitrate content streaming*. US Patent 7,818,444. 2010.

[48] A Zambelli. *Smooth streaming technical overview, microsoft corp., redmond, wa*. Tech. rep. usa, tech. rep, available at http://www. iis. net/learn/media/on-demand-smoothstreaming/smooth-streamingtechnical-overview.

[49] Apple Inc. *HTTP Live Streaming Overview 2013*. [Online]. Available: https://developer.apple.com/library/ios/documentation/networkinginternet/conceptual/streamingmediaguide/Introduction/Introduction.html.

[50] Adobe Systems Inc. *HTTP Dynamic Streaming 2013*. [Online]. Available: http://www.adobe.com/products/hds-dynamic-streaming.html.

[51] European Telecomunications Standard Institute (ETSI) (2009). *Universal Mobile Telecommunication System (UMTS); LTE; Transparent End-to-End Packet Switched Streaming Service (PSS): Protocols and Codecs*. Sophia-Antipolis Cedex, France, 3GPP TS 26.234 Version 9.1.0 Release 9.

[52] DASH Industry Forum. *For Promotion of MPEG-DASH 2013*. [Online]. Available: http://dashif.org.

[53] Guidelines for Implementation: DASH-AVC/264 Interoperability Points. *DASH Industry Forum, 2013*. [Online]. Available: http://dashif.org/w/2013/06/DASH-AVC-264-base-v1.03.pdf.

[54] Ricky K. P. Mok et al. "QDASH: A QoE-aware DASH System". In: *Proceedings of the 3rd Multimedia Systems Conference*. MMSys '12. Chapel Hill, North Carolina: ACM, 2012, pp. 11–22. ISBN: 978-1-4503-1131-1.

[55] Saamer Akhshabi et al. "An experimental evaluation of rate-adaptive video players over HTTP". In: *Signal Processing: Image Communication* 27.4 (2012), pp. 271–287.

[56] Ashkan Sobhani, Abdulsalam Yassine, and Shervin Shirmohammadi. "A fuzzy-based rate adaptation controller for DASH". In: *Proceedings of the 25th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*. ACM. 2015, pp. 31–36.

[57] Guibin Tian and Yong Liu. "Towards agile and smooth video adaptation in dynamic HTTP streaming". In: *Proceedings of the 8th international conference on Emerging networking experiments and technologies*. ACM. 2012, pp. 109–120.

[58] Xiaoqi Yin, Vyas Sekar, and Bruno Sinopoli. "Toward a principled framework to design dynamic adaptive streaming algorithms over http". In: *Proceedings of the 13th ACM Workshop on Hot Topics in Networks*. ACM. 2014, p. 9.

[59] Xiaoqi Yin et al. "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*. ACM. 2015, pp. 325–338.

[60] Truong Cong Thang et al. "An evaluation of bitrate adaptation methods for HTTP live streaming". In: *Selected Areas in Communications, IEEE Journal on* 32.4 (2014), pp. 693–705.

[61] Konstantin Miller et al. "Adaptation algorithm for adaptive streaming over HTTP". In: *Packet Video Workshop (PV), 2012 19th International*. IEEE. 2012, pp. 173–178.

[62] Parikshit Juluri, Venkatesh Tamarapalli, and Deep Medhi. "SARA: Segment Aware Rate Adaptation Algorithm for Dynamic Adaptive Streaming Over HTTP". In: *ICC workshop on QoE-FI* (2015).

[63] Parikshit Juluri, Venkatesh Tamarapalli, and Deep Medhi. "Look-ahead rate adaptation algorithm for DASH under varying network environments". In: *Design of Reliable Communication Networks (DRCN), 2015 11th International Conference on the*. IEEE. 2015, pp. 89–90.

[64] Te-Yuan Huang et al. "A buffer-based approach to rate adaptation: Evidence from a large video streaming service". In: *Proceedings of the 2014 ACM conference on SIGCOMM*. ACM. 2014, pp. 187–198.

[65] Te-Yuan Huang. "A Buffer-Based Approach to Video Rate Adaptation". PhD thesis. Stanford University, 2014.

[66] Xuan Kelvin Zou et al. "Can Accurate Predictions Improve Video Streaming in Cellular Networks?" In: *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*. ACM. 2015, pp. 57–62.

[67] Sami Mekki and Stefan Valentin. "Anticipatory quality adaptation for mobile streaming: Fluent video by channel prediction". In: *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2015 IEEE 16th International Symposium on a*. IEEE. 2015, pp. 1–3.

[68] Wei Bao and Stefan Valentin. "Bitrate adaptation for mobile video streaming based on buffer and channel state". In: *2015 IEEE International Conference on Communications (ICC)*. IEEE. 2015, pp. 3076–3081.

[69] Jiasi Chen et al. "A scheduling framework for adaptive video delivery over cellular networks". In: *Proceedings of the 19th annual international conference on Mobile computing & networking*. ACM. 2013, pp. 389–400.

[70] Junchen Jiang, Vyas Sekar, and Hui Zhang. "Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive". In: *Proceedings of the 8th international conference on Emerging networking experiments and technologies*. ACM. 2012, pp. 97–108.

[71] Vinay Joseph and Gustavo de Veciana. "NOVA: QoE-driven optimization of DASH-based video delivery in networks". In: *INFOCOM, 2014 Proceedings IEEE*. IEEE. 2014, pp. 82–90.

[72] Nicola Bui, Stefan Valentin, and Joerg Widmer. "Anticipatory quality-resource allocation for multi-user mobile video streaming". In: *2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE. 2015, pp. 245–250.

[73] Asit Dan and Don Towsley. *An approximate analysis of the LRU and FIFO buffer replacement schemes*. Vol. 18. 1. ACM, 1990.

[74] Zhe Li, Gwendal Simon, and Annie Gravey. "Caching policies for in-network caching". In: *2012 21st International Conference on Computer Communications and Networks (ICCCN)*. IEEE. 2012, pp. 1–7.

[75] Donghee Lee et al. "LRFU: A spectrum of policies that subsumes the least recently used and least frequently used policies". In: *IEEE transactions on Computers* 50.12 (2001), pp. 1352–1361.

[76] Elizabeth J O'neil, Patrick E O'neil, and Gerhard Weikum. "The LRU-K page replacement algorithm for database disk buffering". In: *ACM SIGMOD Record* 22.2 (1993), pp. 297–306.

[77] Michele Garetto, Emilio Leonardi, and Valentina Martina. "A unified approach to the performance analysis of caching systems". In: *ACM Transactions on Modeling and Performance Evaluation of Computing Systems* 1.3 (2016), p. 12.

[78] Laszlo A. Belady. "A study of replacement algorithms for a virtual-storage computer". In: *IBM Systems journal* 5.2 (1966), pp. 78–101.

[79] Antonio A Rocha et al. "DSCA: A Data Stream Caching Algorithm". In: *Content Caching and Delivery in Wireless Networks, CCDWN, Heidelberg, Germany* (2015).

[80] Salah-Eddine Elayoubi and James Roberts. "Performance and cost effectiveness of caching in mobile access networks". In: *Proceedings of the 2nd International Conference on Information-Centric Networking*. ACM. 2015, pp. 79–88.

[81] Subhabrata Sen, Jennifer Rexford, and Don Towsley. "Proxy prefix caching for multimedia streams". In: *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*. Vol. 3. IEEE. 1999, pp. 1310–1319.

[82] Small Cell Forum. "Backhaul Technologies for Small Cells: Use Cases, Requirements and Solutions". In: *Technical Report/White Paper*. 2013.

[83] Xiaofei Wang et al. "Cache in the air: exploiting content caching and delivery techniques for 5G systems". In: *IEEE Communications Magazine* 52.2 (2014), pp. 131–139.

[84] Mingyue Ji, Giuseppe Caire, and Andreas F Molisch. "Fundamental limits of distributed caching in D2D wireless networks". In: *Information Theory Workshop (ITW), 2013 IEEE*. IEEE. 2013, pp. 1–5.

[85] Francesco Pantisano et al. "Cache-aware user association in backhaul-constrained small cell networks". In: *Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt), 2014 12th International Symposium on*. IEEE. 2014, pp. 37–42.

[86] Georgios Paschos et al. "Wireless caching: Technical misconceptions and business barriers". In: *arXiv preprint arXiv:1602.00173* (2016).

[87] Ejder Bastug, Mehdi Bennis, and Mérouane Debbah. "Living on the edge: The role of proactive caching in 5G wireless networks". In: *IEEE Communications Magazine* 52.8 (2014), pp. 82–89.

[88] Yi Sun et al. "Trace-driven analysis of ICN caching algorithms on video-on-demand workloads". In: *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*. ACM. 2014, pp. 363–376.

[89] Weiwen Zhang et al. "QoE-driven cache management for HTTP adaptive bit rate streaming over wireless networks". In: *IEEE Transactions on Multimedia* 15.6 (2013), pp. 1431–1445.

[90] Yu-Ting Yu et al. "Congestion-aware edge caching for adaptive video streaming in information-centric networks". In: *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*. IEEE. 2015, pp. 588–596.

[91] Andrea Araldo, Fabio Martignon, and Dario Rossi. "Representation selection problem: optimizing video delivery through caching". In: *Networking 2016*. 2016.

[92] Konstantinos Poularakis et al. "Video delivery over heterogeneous cellular networks: Optimizing cost and performance". In: *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. IEEE. 2014, pp. 1078–1086.

[93] Felix Hartanto et al. "Caching video objects: layers vs versions?" In: *Multimedia Tools and Applications* 31.2 (2006), pp. 221–245.

[94] Julien Reichel, Mathias Wien, and Heiko Schwarz. "Scalable video model 3.0". In: *ISO/IEC JTC* 1 (2004).

[95] ISO/IEC JTC1/SC29/WG11. "Information technology - Dynamic adaptive streaming over HTTP (DASH) – Part 1: Media presentation description and segment formats". In: 2012.

[96] Muhammad Zubair Shafiq et al. "Understanding the impact of network dynamics on mobile video user engagement". In: *The 2014 ACM international conference on Measurement and modeling of computer systems*. ACM. 2014, pp. 367–379.

[97] Richard J Kryscio and Claude Lefévre. "On the extinction of the SIS stochastic logistic epidemic". In: *Journal of Applied Probability* (1989), pp. 685–694.

[98] S. Karlin and H.M. Taylor. *A First Course in Stochastic Processes*. First course in stochastic processes / Samuel Karlin; Howard M. Taylor vol. 1. Academic Press, 1975. ISBN: 9780123985521.

[99] P. Balaouras and I. Stavrakakis. "Multimedia Transport Protocols for Wireless Networks". In: *in Emerging Wireless Multimedia: Services and Technologies (eds A. K. Salkintzis and N. Passas)* (2005).

[100] Tobias Hoßfeld et al. "Quantification of YouTube QoE via crowdsourcing". In: *Multimedia (ISM), 2011 IEEE International Symposium on*. IEEE. 2011, pp. 494–499.

[101] ISO/IEC IS 23009-1. "Dynamic adaptive streaming over HTTP (DASH) - Part 1: Media presentation description and segment formats". In: 2012.

[102] Thomas Stockhammer. "Dynamic adaptive streaming over HTTP–: standards and design principles". In: *Proceedings of the second annual ACM conference on Multimedia systems*. ACM. 2011, pp. 133–144.

[103] Saamer Akhshabi, Ali C Begen, and Constantine Dovrolis. "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP". In: *Proceedings of the second annual ACM conference on Multimedia systems*. ACM. 2011, pp. 157–168.

[104] Christopher Müller, Stefan Lederer, and Christian Timmerer. "An evaluation of dynamic adaptive streaming over HTTP in vehicular environments". In: *Proceedings of the 4th Workshop on Mobile Video*. ACM. 2012, pp. 37–42.

[105] Haakon Riiser et al. "Commute Path Bandwidth Traces from 3G Networks: Analysis and Applications". In: *Proceedings of the 4th ACM Multimedia Systems Conference*. MMSys '13. Oslo, Norway: ACM, 2013, pp. 114–118.

[106] Michael Grafl et al. "Scalable video coding guidelines and performance evaluations for adaptive media delivery of high definition content". In: *2013 IEEE Symposium on Computers and Communications (ISCC)*. IEEE. 2013, pp. 000855–000861.

[107] Michael Grafl et al. "Evaluation of hybrid scalable video coding for HTTP-based adaptive media streaming with high-definition content". In: *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013 IEEE 14th International Symposium and Workshops on a*. IEEE. 2013, pp. 1–7.

Résumé étendu en français

Chapitre 1
Introduction

Malgré les récentes évolutions dans les réseaux de télécommunications, la transmission de la vidéo streaming sur les réseaux mobiles est toujours problématique. La raison principale étant l'accroissement fulgurante de contenu vidéo ces dernières années. Selon les rapports de Cisco sur l'évolution des données mobiles, la quantité de données mobiles mensuelles atteindra 30.6 exaoctets d'ici 2020. Trois quarts (3/4) de ces données mobiles seront représentatifs de contenu vidéo (Fig. 7.1).

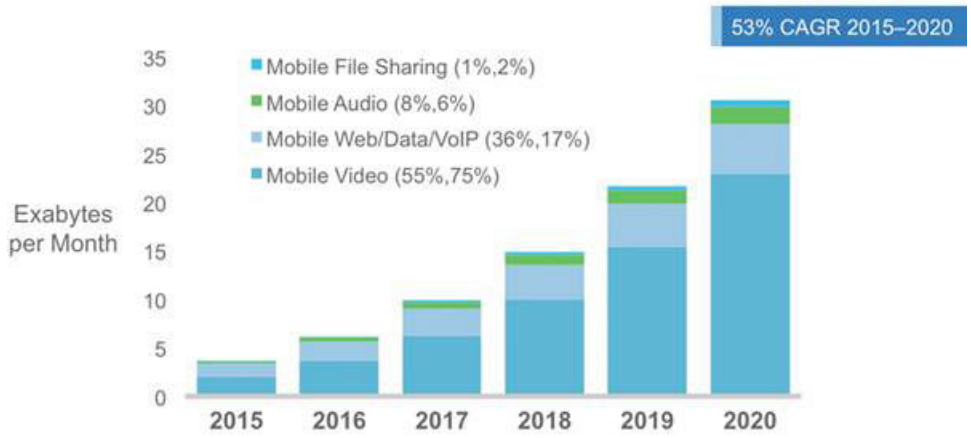

FIGURE 7.1: Quantité totale de différents trafic par mois en exaoctets entre 2015 et 2020 par Cisco.

Parmi les récentes évolutions dans les réseaux de télécommunications, on retrouve la norme de 4^{eme} génération ($4G$) LTE et la $5G$ toujours en cours de standardisation. La $4G$ LTE est le successeur des normes $3G$ normalisé par l'organisme $3GPP$. Elle offre de meilleures performances par rapport aux générations antérieures: un débit allant jusqu'à 1Gbps, une éfficacité de la bande passante aussi bien pour le lien montant que le lien descendant (30 bits/s/hz), de temps de latence moindre (< 100 ms pour le plan de contrôle et < 10 ms pour le plan utilisateur), une plus grande capacité de cellule, etc. En plus de cela, il y a l'interopérabilité avec les technologies existantes. La norme LTE a été améliorée à travers la LTE-Advanced ou norme LTE avancée. C'est la même architecture que la norme LTE mais qui utilise en plus des techniques pour accroitre les débits utilisateurs. Parmi lesquelles l'agrégation de porteuses, la technique d'antennes multiples pour prendre avantage des effets du multi-trajet, les noeuds relais, ou encore les techniques de multipoint coordonné, etc. La technologie $5G$ quant à elle, est en cours de standardisation. Elle devrait être une combinaison des technologies existantes couvrant la $2G$, $3G$, $4G$, $Wi-Fi$ et d'autres technologies. Elle devrait utiliser de nouvelles techniques comme la technique de modulation ou les techniques d'accès au lien

radio. La norme $5G$ permettra d'offrir des débits utilisateurs supérieurs à 1Gbps, une plus grande capacité, des temps de latence encore plus petits (< 1 ms), un plus grand nombre d'utilisateurs, 100% de couverture, 90% de réduction en énergie utilisée, etc. Parmi les services offerts, on trouve la réalité virtuelle, la réalité augmentée, la conduite autonome, les objets connectés, etc.

En plus de l'accroissement du contenu vidéo sur Internet, les exigences des utilisateurs en terme de qualité ont accrues. Ainsi ils préfèrent de plus en plus visionner la vidéo en haute qualité ou même ultra haute qualité par exemple les résolutions $4K$ ou $8K$. Donc malgré les débits importants offerts par les nouvelles technologies de réseaux telle que la $4G$ LTE, la transmission de contenu vidéo reste toujours aussi problématique. La vidéo streaming utilise plusieurs techniques.

L'élément principal dans le concept de la vidéo streaming est la possibilité de pouvoir visionner la vidéo avant qu'elle ne soit complètement téléchargée. Ainsi le concept de streaming se distingue du téléchargement classique où il faut attendre d'avoir le fichier complet afin de pouvoir visionner le contenu. La technologie de la vidéo streaming est divisée en deux catégories: la vidéo à la demande où le contenu est préalablement stocké sur un serveur (Youtube par exemple) et le streaming temps réel (par exemple un évènement de football en direct). On s'intéresse à la vidéo à la demande dans cette thèse. Le streaming englobe plusieurs protocoles à différents niveaux du modèle de référence OSI parmi lesquels les protocoles de transport TCP et UDP. Le premier étant plus fiable car permettant la retransmission de paquets. Parmi les technologies de streaming utilisées, on retrouve RTP, le téléchargement progressif, le streaming adaptatif.

Le RTP ou protocole de transport temps-réel utilise UDP. Il est associé avec le protocole RTCP pour le contrôle de la session. RTP est utilisé par Flash, RealPlayer, le lecteur Windows Media ou encore le lecteur QuickTime.

Le téléchargement progressif n'est pas lié à un protocole particulier. Il suit le concept streaming et on n'a pas la possibilité de changer la qualité de la vidéo pendant la session.

Le streaming adaptatif, quant à lui donne la possibilité à l'utilisateur de changer la qualité de la vidéo pendant la session. Ainsi on peut adapter la qualité de la vidéo en fonction des conditions du réseau. C'est une technologie qui utilise les protocoles HTTP et TCP. L'avantage de l'utilisation du protocole HTTP est de se passer des contraintes de blocage liées aux parefeux et aux translations d'adresses. Avec le streaming adaptatif, la vidéo doit être disponible en plusieurs qualités, le fichier associé à chaque qualité est appelé une représentation. Le fichier original doit être codé pour donner les représentations. C'est le codage de la vidéo. L'outil responsable du codage et décodage de la vidéo est le *codec*. Le streaming adaptatif a pu devenir populaire grâce à l'efficacité des codecs. Dans cette thèse, on se focalise sur l'utilisation du codec SVC. Le streaming adaptatif a été normalisé sous le nom de DASH par l'UIT et le groupe MPEG.

Objectifs

L'objectif principal de cette thse est d'améliorer la performance des services de vidéo streaming dans les réseaux mobiles. Nous procédons par deux approches.

La première est la conception d'une solution complémentaire au standard DASH appelé BSC (Backward-Shifted Coding) basé sur le codec SVC. Dans le standard DASH, la manière dont est faite l'adaptation de la qualité de la vidéo n'est pas normalisée, le choix est laissé aux fournisseurs de contenu. Notre solution proposée, BSC, est plus qu'un simple algorithme d'adaptation. Il est moins couteux car n'engendre pas de modifications de l'architecture de DASH. Nous analysons les performances de ce nouveau

système à travers une fonction de coût de la qualité d'expérience. Cette fonction implique les métriques suivantes: la probabilité de blocage de la vidéo, le nombre total de blocages, le temps d'attente initial et la qualité moyenne de la vidéo. Finalement nous concevons des algorithmes d'adaptation pour le nouveau système BSC.

La deuxième approche est basée sur le caching de la vidéo streaming. Avec le caching, le contenu est plus proche des utilisateurs, ce qui permet de réduire les temps de latence et d'améliorer l'expérience utilisateur. Cependant, le paradigme du caching selon lequel une vidéo est un objet unique n'est plus valable dans le cas du streaming adaptatif. En effet, parce qu'il est possible de changer de qualité de la vidéo pendant la session, plusieurs copies de la même vidéo sont disponibles. Ces copies sont appelées des représentations. Ainsi d'un segment à un autre on peut changer la qualité en téléchargeant la bonne représentation de cette vidéo. Dans le caching, il devient dont nécessaire de concevoir des algorithmes qui prennent en compte la qualité de la vidéo à mettre en cache. Nous proposons ces nouveaux types d'algorithmes dans la deuxième partie de la thèse.

Contributions

Ici, nous listons les contributions de la thèse. Tout d'abord nous montrons les résultats de deux travaux qui n'ont pas été inclus dans les différentes parties à venir. Il s'agit de nos travaux sur l'analyse du trafic vidéo streaming dans les réseaux 4G LTE puis de l'analyse et la modélisation de la qualité d'expérience dans un modèle fluide de chaines de Markov. Ensuite, nous citons les contributions par chapitre.

Nous avons évalué le trafic de vidéo streaming dans **[C1]**. Nous avons utilisé un simulateur de réseaux niveau système développé par l'université de Vienne qui permet d'analyser les fonctionnalités d'un réseau telles que l'ordonnancement des ressources utilisateurs, la gestion de la mobilité, la gestion des interférences et la propagation des signaux. En considérant plusieurs scénarios de simulations, nous avons analysé la gigue (inter-arrivée) des trames vidéos. En effet, les trames vidéos sont envoyées de manière régulière au niveau du serveur vidéo mais arrivent au niveau des utilisateurs d'une manière irrégulière à cause des conditions canal du réseau. Nous avons tracé la distribution de la gigue et les distributions standard qui coïncident comme le montre la Fig. 7.2. Les résultats ont montré l'efficacité des réseaux 4G LTE pour les services gourmand en bande passante comme la vidéo streaming.

Nous avons étudié les métriques de la qualité d'expérience en utilisant les modèles fluides de chaines de Markov dans **[C2]**. Ceci étant fait sans contraintes sur la taille de la vidéo comme de coutume. La taille de la vidéo est donc N trames. Le réseau est modélisé comme une chaine de chaine de Markov à modèle fluide. Nous dérivons des équations différentielles partielles et nous utilisons une méthode de double transformée de Laplace pour résoudre ces équations. La méthode utilisée est une méthode polynomiale inspirée des distributions du premier temps de passage (à zéro) de [36]. Une première transformée de Laplace sur l'équation différentielle partielle donne une équation différentielle ordinaire. Cette équation différentielle ordinaire est résolue en utilisant le determinant de l'équation polynomiale pour obtenir la transformée de Laplace de la solution. Ainsi, une transformée de Laplace inverse est utilisée sur la dernière solution en utilisant les méthodes d'inversion de [37, 38].

Nous citons maintenant les contributions par chapitre: Dans le deuxième chapitre, nous présentons l'état de l'art du streaming adaptatif et du caching. Nous commençons par décrire l'architecture du système étudié dans la thèse. Nous parlons aussi des codecs utilités pour la compression de la vidéo. Ensuite, nous relatons les notions à propos de la qualité d'expérience, elle sera utilisée plus tard pour l'analyse de performances du

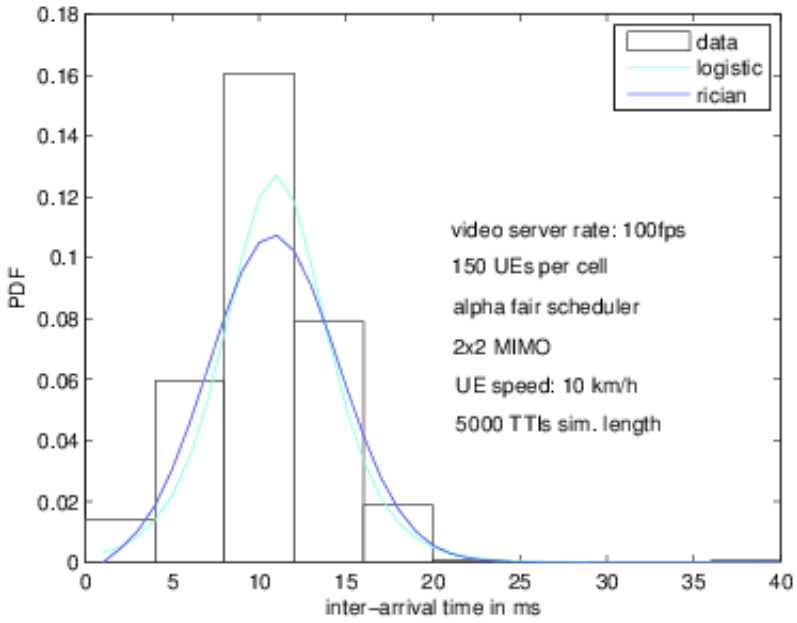

FIGURE 7.2: Fonction de densité de masse de la gigue des trames vidéo.

système BSC. Nous montrons aussi les algorithmes d'adaptation existants dans la littérature DASH. Finalement, nous passons en revue les algorithmes de caching existants et pourquoi ils ne sont pas satisfaisants.

Dans le troisième chapitre, nous expliquons le système BSC. Nous commençons par décrire en détails le codec SVC car le système BSC est entièrement basé sur ce codec. Ensuite nous donnons réellement l'idée principale du BSC, comment marche le BSC? et comment l'incorporer dans le standard DASH. Finalement nous discutons de l'implémentation du protocole BSC dans DASH. Les contributions de ce chapitre se trouve dans **[C3]** et **[W6]**.

Dans le chapitre 4, nous analysons les performances du système BSC. Pour cela, on se base sur la qualité d'expérience car elle est préférée à la qualité de service pour l'analyse des systèmes complexes comme ceux de la vidéo streaming. Dans notre analyse, on calcule la probabilité que le tampon du lecteur soit vide (blocage) et la probabilité d'avoir plusieurs blocages en utilisant le théorème de Ballot. Puis nous calculons le temps d'attente initial et la qualité moyenne de la vidéo en utilisant la distribution quasi stationnaire des processus de mort-naissance. Finalement, nous proposons une fonction objective de la qualité d'expérience en utilisant les métriques calculées précédemment. Les résultats de l'analyse du BSC se trouvent dans **[C3]**.

Dans le chapitre 5, nous concevons des algorithmes d'adaptation à utiliser dans le système BSC. Nous commençons par l'intégration de BSC dans DASH. Dans DASH, la vidéo est divisée en petits morceaux appelés segments, chacun étant disponible en plusieurs qualités. Le système BSC consiste à transmettre deux segments superposés: un segment basse couche et un segment haute couche. La principale contribution de ce chapitre est la sélection de la qualité des deux segments superposés étant donné la capacité du réseau disponible. Ceci afin de minimiser les interruptions de la vidéo, d'accroitre la qualité de la vidéo et de diminuer les variations de qualité. Finalement, on utilise des simulations avec des traces de débit synthétique et empirique afin de comparer les résultats de BSC avec ceux de DASH classique. Les contributions de cette

partie concernant l'adaptation sont publiées dans **[C4]**.

Dans le chapitre 6, nous étudions le problème de caching dans les systèmes de streaming adaptatif. En effet le caching dans le réseau d'accès (macro cellules ou petites cellules) permet de rapprocher le contenu des utilisateurs afin de décharger les liens du réseau cœur. Avec le streaming adaptatif chaque vidéo est disponible en plusieurs copies. Dans ce chapitre, nous étudions des politiques de caching qui permettent d'optimiser le contenu du cache. Pour analyser la performance des algorithmes de caching proposés, nous cherchons la solution optimale de caching en utilisant la programmation linéaire afin de minimiser la taille de contenu téléchargé depuis le serveur distant. Finalement, des simulations permettrons de comparer les performances des algorithmes de caching proposés avec ceux déjà existants. Les résultats de cette partie se trouvent dans **[S7]**.

Dans le chapitre 7, nous résumons les résultats des chapitres précédents sur l'analyse de performances du système BSC, les algorithmes d'adaptation dans le système BSC et le caching dans les systèmes de streaming adaptatif. Ensuite nous discutons des perspectives de la thèse sur la gestion de la qualité d'expérience, le streaming adaptatif et le caching.

Publications

La liste des publications est ci-dessous.

Conférences internationales:

[C1]: **Zakaria Ye**, Tania Jiménez and Rachid El-Azouzi."Video streaming analysis in Vienna LTE system level simulator." *Proceedings of the 8th International Conference on Simulation Tools and Techniques*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), August 2015, Athens, Greece.

[C2]: **Zakaria Ye**, Rachid EL-Azouzi and Tania Jiménez. "Analysis and modelling Quality of Experience of video streaming under time-varying bandwidth." *Wireless and Mobile Networking Conference (WMNC)*, 2016 9th IFIP. IEEE, July 2016, Colmar, France.

[C3]: **Zakaria Ye**, Rachid EL-Azouzi, Tania Jiménez, Eitan Altman and Stefan Valentin. "Backward-Shifted Strategies Based on SVC for HTTP Adaptive Video Streaming." *Proceedings of the 15th International IFIP Networking Conference*, IFIP, May 2016, Vienna, Austria.

[C4]: **Zakaria Ye**, Rachid EL-Azouzi, Tania Jiménez, Francesco De Pellegrini and Stefan Valentin. "Bitrate Adaptation in Backward-Shifted Coding for HTTP Adaptive Video Streaming." *In IEEE International Conference on Communications (ICC)*, May 2017, Paris, France.

Workshops:

[W5]: **Zakaria Ye**, Rachid EL-Azouzi, Tania Jiménez and Eitan Altman. "A Queueing Theoretic Approach to Design a Backward-Shifted Strategies Based on AVC/SVC for HTTP Adaptive Streaming." Poster Presentation, *1st GdR MaDICS Workshop on Big Data for the 5G RAN*, November 2015, Paris, France.

[W6]: **Zakaria Ye**, Rachid EL-Azouzi, Tania Jiménez and Stefan Valentin. "Backward-Shifted Coding (BSC) for HTTP Adaptive Streaming." *11ème Atelier en Evaluation de Performances*, du 15 au 17 mars 2016, Toulouse, France.

Articles soumis:

[S7]: **Zakaria Ye**, Francesco De Pellegrini, Rachid EL-Azouzi and Tania Jiménez. "Rate Aware Video Caching Schemes for 5G Networks." *Submitted to ITC*, September 2017, Genova, Italy.

Chapitre 2
Etat de l'art et contexte

Dans ce chapitre, nous présentons l'état de l'art de notre travail sur l'analyse de performance des services de vidéo streaming adaptatif. Pour se faire, nous parlons du standard streaming adaptatif sur HTTP et du caching de la vidéo. Nous commençons par présenter l'architecture type d'un système de vidéo streaming, puis les codecs vidéo et les notions de la qualité d'expérience. Ensuite nous citons une liste non exhaustive des algorithmes d'adaptation dans DASH pour finir avec les stratégies de caching existantes.

Architecture des systèmes vidéo streaming

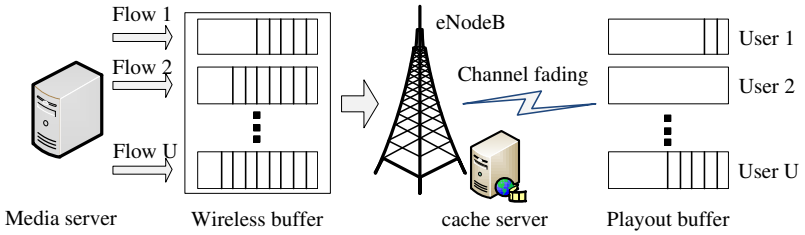

FIGURE 7.3: Architecture d'un système vidéo streaming

La Fig. 7.3 montre l'architecture typique d'un système de vidéo streaming. Les composants essentiels de ce système sont le réseau, le serveur vidéo, les clients et les caches déployés dans le réseau. Le rôle principal du réseau est de permettre la transmission d'un fichier vidéo du serveur vers le terminal de l'utilisateur final. Toutes les vidéos sont stockées dans le serveur vidéo. Le rôle du cache est de permettre de stocker certaines vidéos. Lorsque ces vidéos sont demandées il n y a plus besoin de les transmettre depuis le serveur, ce qui va réduire considérablement les temps de transmission.

Codecs vidéo

Le rôle d'un codec est de compresser une source vidéo pour faciliter la transmission de cette dernière. Une liste non exhaustive des codecs les plus populaires sont H264/AVC, H264/SVC, H264/MVC, RTV, H265/HEVC, VP8, etc. Un codec est caractérisé par entre autres, l'efficacité de compression, la complexité du codage décodage, la robustesse au erreurs et pertes, le délai de bout en bout. H264/AVC, créé en 2003, est le codec le plus utilisé de nos jours. Il résulte de l'évolution des codecs standards de l'UIT depuis 1990. Il a été normalisé par l'UIT et le groupe multimédia MPEG. C'est un codec mono couche utilisé par YouTube, la boutique iTunes, Le lecteur Flash d'Adobe, Microsoft Silverlight, etc. Il contient 21 profils ou extensions parmi lesquels on peut citer le codage à multi vues MVC (pour la 3D) et le codage multi couches SVC. H265/HEVC est l'évolution récente de AVC qui le remplacera dans les années à venir. Introduit en janvier 2013, il devrait être deux fois plus performant que le codec AVC capable de rendre les vidéos avec des résolutions 4K (2160p), 8K (4320p) ou même au delà. RTV est une version propriétaire de Microsoft qui intègre des codes correcteurs d'erreurs et VP8 est une solution open source utilisée par Skype pour les appels vidéo.

Qualité d'expérience de la vidéo streaming

La qualité d'expérience est une notion subjective qui permet de mesurer la satisfaction de l'utilisateur d'un service. Avec la complexité des systèmes de nos jours, la qualité de service est de moins en moins utilisée car une bonne qualité de service ne garantit pas obligatoirement une bonne expérience des utilisateurs vis à vis du service. La qualité d'expérience est une notion qui est influencée par le système, les utilisateurs et le contexte. Le système, parce qu'elle est liée aux paramètres du système (en l'occurrence ceux de la qualité de service), les utilisateurs n'ont pas les mêmes besoins et le contexte est souvent différent, par exemple du football en direct ou une série télévisée en différé. La qualité d'expérience a besoin souvent d'être quantifiée pour pouvoir la prédire et la contrôler.

On rencontre alors les métriques suivantes dans la littérature:

- **Le délai initial:** C'est le temps de latence entre le moment où on initie la session vidéo et le moment où la vidéo commence à jouer. Pendant ce temps, le lecteur accumule une certaine quantité d'images dans le tampon. Il en est de même après chaque blocage de la vidéo.
- **Le ratio de blocage:** Lorsque le tampon du lecteur est vide (il n y a plus d'images à lire), le lecteur se bloque. C'est un blocage ou une interruption. Le ratio de blocage est la proportion de temps que le lecteur passe dans le blocage au temps de lecture.
- **Le taux de blocage:** C'est la fréquence à laquelle les blocages interviennent pendant la session. C'est le ratio entre le nombre total de blocage et la durée de la vidéo.
- **La qualité moyenne:** La qualité de la vidéo est le débit binaire en kilobits par seconde. Dans le cas d'un débit binaire variable, la qualité moyenne est le débit binaire moyen.
- **Le nombre de variations de qualité:** C'est le nombre de variations de qualité pendant la session.

En plus de ces métriques interdépendantes qui représentent la qualité d'expérience, certains facteurs externes influencent la qualité perçue par l'utilisateur. Ainsi, quantifier la qualité d'expérience en une entité unique est très complexe.

Mesurer la qualité d'expérience est donc complexe. Parmi les méthodes de mesure, on a la qualité de service, le MOS. En ce qui concerne la QoS, il faut trouver une fonction entre la qualité d'expérience et les paramètres de la qualité de service. Le MOS permet de quantifier la satisfaction de l'utilisateur sur une échelle de 1 à 5 par exemple. Ces méthodes sont assez contraignantes car elle nécessitent la participation de l'utilisateur où même du système. De nouvelles méthodes de mesure utilisent ce qu'on appelle l'engagement de l'utilisateur. Il peut s'agir du temps de visionnage de la vidéo, du nombre de visites sur la plateforme ou le site, de la probabilité que l'utilisateur retourne sur le site [23, 24]. Il est facile de mesurer cette entité qu'est l'engagement au niveau du terminal de l'utilisateur sans même son intervention. Ainsi dans la littérature, plusieurs études [23, 24, 25, 26] se sont basées sur l'obtention d'une fonction du type

$$Engagement = f(QualityMetric_i) \tag{7.1}$$

Les facteurs externes suivants ont été identifiés: la nature de la vidéo (vidéo temps réel, vidéo à la demande), le type d'appareils (téléphones portables, PCs, télévisions), la connectivité (cable, réseaux sans fil), les besoins personnels des utilisateurs.

Plusieurs études analytiques de la qualité d'expérience ont été effectuées. Il s'agit d'une part de calculer les métriques ci-dessus et d'autre part de trouver une fonction pour quantifier la qualité d'expérience à travers ces métriques. Les méthodes utilisées pour cela sont principalement basées sur l'analyse du régime transitoire des files d'attente. Un processus d'arrivée, e.g, exponentiel, modélise le réseau, le processus de départ est fonction de la fréquence de lecture et le système étudié est le tampon du lecteur. On a la méthode utilisant les équations différentielles partielles et ordinaires, les files d'attente de type G/G/1, ou encore M/M/1 ou M/D/1 utilisant le théorème du scrutin.

Le streaming adaptatif sur HTTP

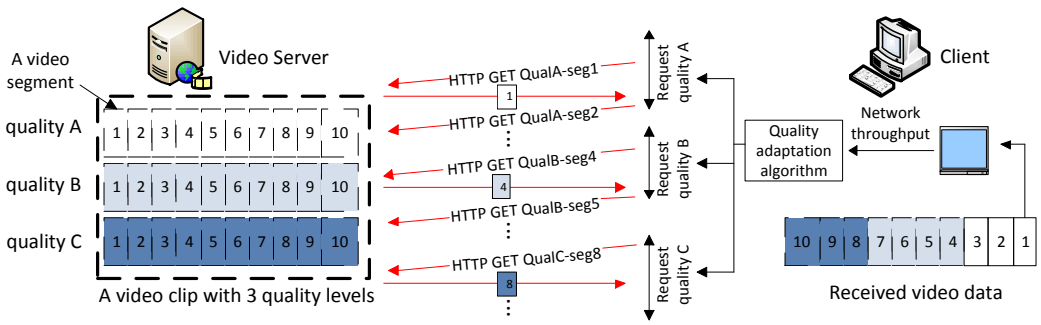

FIGURE 7.4: Architecture du système DASH [54]

Le streaming adaptatif a vu le jour avec les solutions propriétaires de Microsoft, Apple et Adobe, respectivement en 2008, 2009 et 2010. La première version du standard est apparue en 2009 pour être finaliser sous l'appellation DASH (Fig. 7.4) en 2012. De nos jours d'autres solutions propriétaires existent chez Netflix, Akamai, Movestreaming ou encore Amazon. DASH utilise le protocole HTTP sur TCP, ce qui lui permet de ne pas être bloqué par les pare-feux et la translation d'adresses.

Le principe est le suivant. Le fichier vidéo est divisé en plusieurs segments de durée fixe. Chaque segment est disponible en plusieurs qualités appelées représentations. Les segments sont stockés sur le serveur vidéo et téléchargés à la demande de l'utilisateur à travers des requêtes GET HTTP. Ainsi l'utilisateur peut à chaque fois télécharger la qualité correspondante à sa connexion réseau. On parle de l'adaptation de la qualité. La manière de faire cette adaptation n'est pas normalisée, le but de l'adaptation étant simplement d'éviter le blocage du lecteur, d'augmenter la qualité de la vidéo et aussi de minimiser le changement de qualité qui peut perturber la vidéo.

L'adaptation se fait de la manière suivante (voir Fig. 7.5). Après le téléchargement d'un segment, on se base sur la qualité de la connexion pour télécharger le prochain segment. Il existe énormément d'algorithmes d'adaptation dans le littérature. Ils se basent soit sur l'estimation de la qualité du lien, soit sur la quantité de données dans le tampon du lecteur, soit sur les deux pour décider de la qualité du prochain segment.

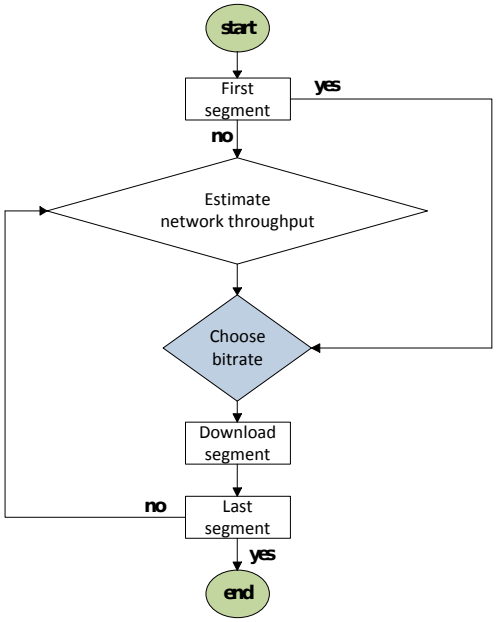

FIGURE 7.5: Diagramme d'adaptation de la qualité pour DASH

Caching de la vidéo

Le caching permet de rapprocher certain contenu des utilisateurs afin de réduire le trafic backhaul du réseau. Les caches sont limités en taille, et on ne peut donc pas stocker la totalité des vidéos. On parle alors de politique de caching qui permet de gérer le cache: quelle vidéo doit être mise en cache, quelle autre vidéo doit sortir du cache? Il existe plusieurs politiques de caching dans la littérature.

LRU: Toute nouvelle requête est placée en haut du cache, qu'elle existe déjà ou non dans le cache. Lorsque qu'il n y a pas de place pour une nouvelle requête, le contenu en bas du cache est supprimé (Fig. 7.6).

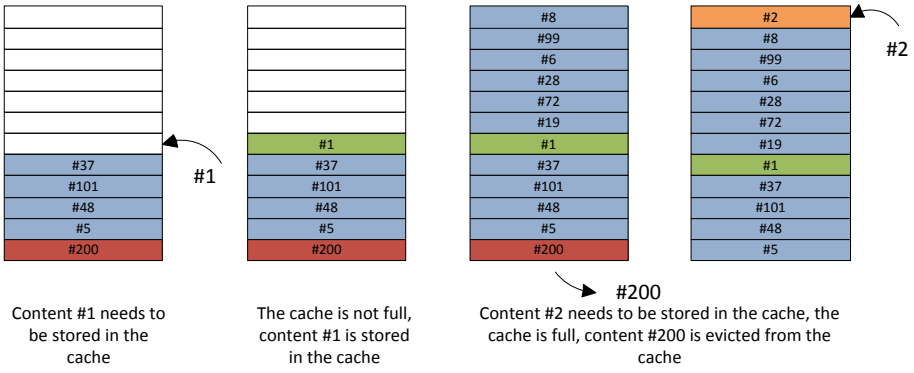

FIGURE 7.6: L'algorithme de caching LRU

LFU: Chaque vidéo se verra incrémenter un compteur à chaque nouvelle requête. L'algorithme garde dans le cache les vidéos ayant le plus de requêtes.

FIFO: Il est comme l'algorithme LRU. La seule différence est que lorsqu'une requête arrive pour une vidéo qui existe déjà dans le cache, sa position n'est pas mise à jour.

Random: Il est identique à LRU pour l'insertion d'une vidéo dans le cache. Par contre l'éviction se fait de manière aléatoire.

LRFU: C'est un spectre d'algorithmes entre LRU et LFU. Il est gouverné par un paramètre compris entre 0 et 1. Les bornes correspondent à LRU et LFU.

LRU-k: Pour chaque vidéo il enregistre les dernières k fois qu'une a été demandée et supprime du cache la vidéo avec le plus petit k^{eme} accès.

k-LRU: Chaque vidéo passe dans $k-1$ caches virtuels avant qu'elle ne soit stockée dans le cache physique. Chaque cache virtuel suit la politique LRU.

Algorithme de Belady: Pour faire de l'espace dans le cache pour une nouvelle vidéo, cet algorithme supprime le contenu qui a le temps de prochaine requête le plus long. Pour cela on doit connaitre le future des requêtes. Ce qui n'est pas possible en pratique. Il n'est pas implémentable mais il permet d'avoir la solution optimale pour comparer les autres algorithmes de caching.

Ces algorithmes de caching ne sont pas adaptés dans le cadre du streaming adaptatif car ils permettent de savoir la vidéo à mettre dans le cache mais ne sont pas capable de dire quelle qualité. En effet, dans le streaming adaptatif, chaque vidéo est disponible en plusieurs copies, chaque copie correspondant à une qualité. Le problème a commencé à être étudié. Parmi les politiques proposées utilisant LRU on a *allQ* (stocker toutes les qualités à la limite de l'espace disponible), *onlyLQ* (stocker que la plus basse qualité pour chaque vidéo), *onlyHQ* (stocker que la plus haute qualité pour chaque vidéo), *Qimpr* (commencer par la basse qualité et augmenter la qualité à chaque nouvelle requête), *Partitionned* (stocker le même nombre de vidéos pour chaque qualité dans le cache). D'une part ces politiques sont beaucoup trop simplistes et d'autre part ils fonctionnent avec le codec AVC mais pas avec SVC.

Dans le prochain chapitre nous présentons l'architecture du protocole de transmission BSC.

Chapitre 3
Backward-Shifted Coding

Dans ce chapitre nous expliquons ce qu'est le protocole BSC. On commence par expliquer un peu plus en détails le codec SVC avant de dire comment marche BSC et comment on l'intégrera et l'implémentera dans le standard DASH.

Scalable Video Coding

Le codec SVC est une extension du codec AVC introduit en 2007. Il est un codec multi-couches. La vidéo est codée en plusieurs couches, une couche basique et une ou plusieurs couches d'amélioration. Lorsque seule la couche basique est utilisée, la vidéo est rendue avec la qualité minimale. A chaque ajout d'une couche d'amélioration à la couche basique et aux autres couches d'amélioration, on augmente la qualité de la vidéo. Les couches d'amélioration agissent sur le taux d'images par seconde, la résolution de l'image mais aussi la fidélité de l'image. Comme montré dans la Fig. 7.7, chaque couche d'amélioration est dépendante de la couche basique et de la couche d'amélioration précédente. Un avantage important de SVC est que c'est un codec moins

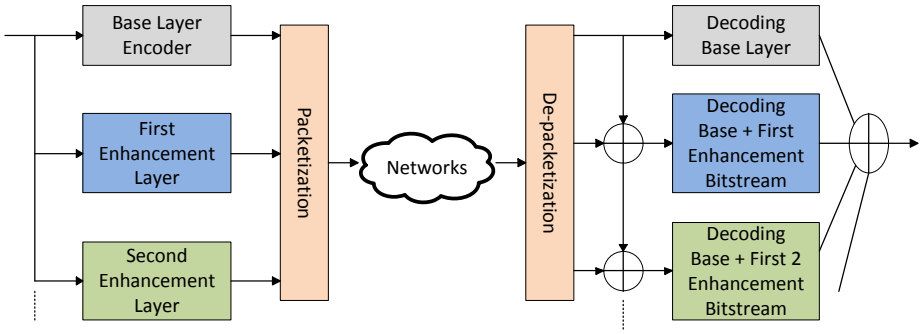

FIGURE 7.7: Codage et décodage avec le codec SVC

sensible aux erreurs dans le réseau telles que les pertes de paquets.

Lorsqu'il est utilisé dans DASH, SVC permet à l'utilisateur de télécharger chaque segment avec la qualité adéquate en téléchargeant le nombre de couches nécessaires pour arriver à cette qualité. Dans l'exemple de la Fig. 7.8, on voit que le client

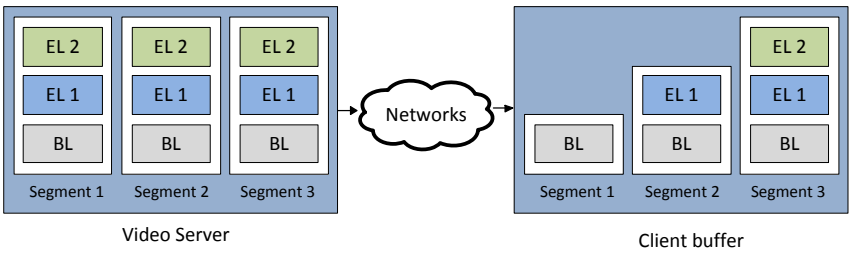

FIGURE 7.8: SVC dans DASH

télécharge le premier segment avec la qualité minimale, le deuxième segment avec la qualité intermédiaire et le dernier segment avec la qualité maximale.

Backward-Shifted Coding (BSC)

Nous expliquons dans cette section, comment marche le protocole BSC au niveau images ensuite au niveau segments.

Supposons un fichier vidéo de N images numérotées de 1 à N. Supposons que le codec SVC a rendu la vidéo avec deux couches après codage (une couche basique et une couche d'amélioration). On peut généraliser à plus de deux couches, dans ce cas on considérera la couche basique et l'ensemble des couches d'amélioration comme une seule couche, ce qui nous donnera encore le cas de deux couches. L'idée de base de BSC est de transmettre la couche basique et la couche d'amélioration d'une même image séparément comme le montre la Fig. 7.9. La couche basique est envoyée avant la couche

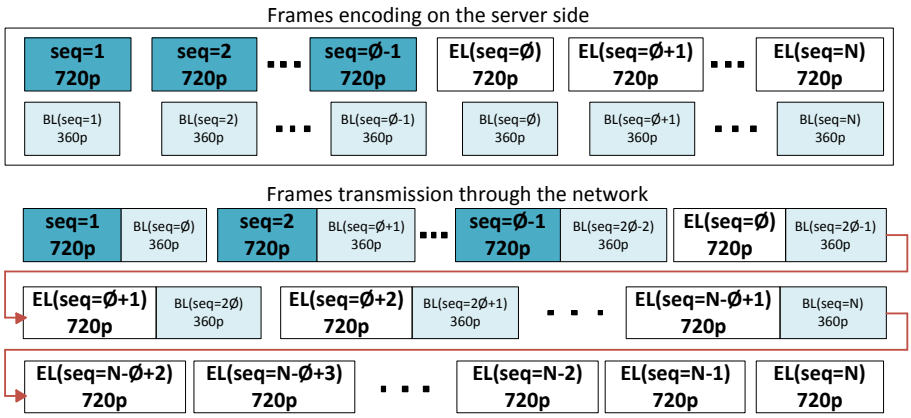

FIGURE 7.9: Le schéma BSC au niveau images

d'amélioration. On fait ceci pour toutes les images. Ainsi si la couche d'amélioration est envoyée dans l'image k alors la couche basique correspondante a déjà été envoyée dans l'image $k-\phi+1$. Le paramètre ϕ est très important dans le schéma BSC, il correspond au décalage introduit entre la couche basique et la couche d'amélioration d'une image.

Au niveau client, lorsque la couche d'amélioration d'une image n'est pas arrivée à temps, la couche basique pourra être utilisée seule avec une qualité moindre certes mais on pourra éviter le blocage de la vidéo.

Pour l'intégration de BSC dans DASH, on se place au niveau segments car le standard DASH fonctionne de la sorte. En effet la vidéo pourra être divisée en K segments numérotés de 1 à K. Chaque segment est codé avec SVC pour donner plusieurs couches donc plusieurs qualités comme le montre la Fig. 7.10. On a cinq qualités différentes à savoir 144p, 240p, 360p, 480p et 720p. La couche basique correspond à la qualité de 144p tandis que toutes les couches réunies correspond à la qualité de 720p. Comme dit précédemment, dans le standard DASH, chaque segment est demandé avec une qualité donnée correspondante au débit réseau, ce qui revient à envoyer la couche basique avec les couches d'amélioration nécessaires pour atteindre cette qualité. Dans BSC, à l'instant t_k, le segment k est demandé avec une qualité donnée (couche basique plus quelques couches d'amélioration). Á l'instant $t_{k+\phi-1}$, quelques couches d'amélioration

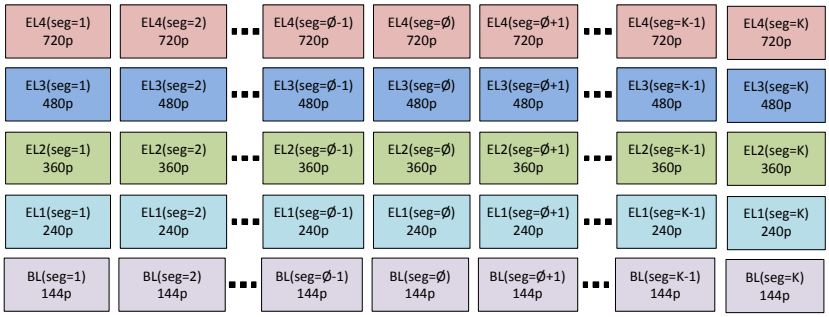

FIGURE 7.10: Les segments vidéo codés avec SVC

du segment k sont transmises dans le segment $k+\phi-1$ dans le but d'améliorer la qualité précédente (Fig. 7.11). BSC transmet donc un segment complet avec les couches

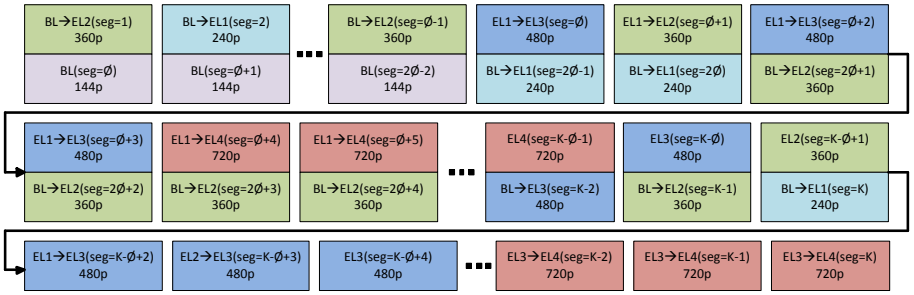

FIGURE 7.11: Transmission des segments avec BSC: le segment de couche basse contient une couche basique (plus quelques couches d'amélioration)et est transmis avant le segment de couche haute correspondant qui suivra après $\phi-1$ blocs; les $\phi-1$ blocs initiaux transportes uniquement des segments de couche basse; la notation $BL \rightarrow EL_j$ indique tous les segments BL, EL_1, EL_2, ..., EL_j et $EL_i \rightarrow EL_j$ indique EL_i, EL_{i+1}, ..., EL_j

d'amélioration d'un autre segment. On appelle le premier segment, segment de couche basse et le deuxième est le segment de couche haute. Le paramètre ϕ est appelé le décalage entre le segment de couche basse et le segment de couche haute. Ainsi chaque segment k a des couches d'amélioration transmises avec le segment $k+\phi-1$. On appelle bloc k, la combinaison du segment $k+\phi-1$ et des couches d'amélioration du segment k.

Au niveau client, les paquets réseaux sont rassemblés en images par le décodeur. La couche basse et la couche haute d'un même segment sont transmises séparément, le décodage aura lieu deux fois, le premier décodage est fait lorsque la couche basse est transmise pour rendre le segment avec la qualité correspondante, le deuxième décodage est fait lorsque la couche haute sera transmise après $\phi-1$ segments. Cette couche haute permettra d'améliorer la qualité du segment.

Implémentation de BSC

Pour faciliter la compréhension de BSC, il faut connaitre la structure du fichier MPD. C'est un fichier XML qui décrit toutes les caractéristiques de la vidéo. Il contient

des informations à propos de chaque segment vidéo, les relations entre les segments et d'autres méta-données pour une bonne synchronisation entre le client et le serveur. Il est échangé entre le client et le serveur au tout début de la transmission.

L'étude du fichier MPD nous a montré que le protocole BSC ne nécessite aucunes modifications au niveau du serveur vidéo. Le plus important aspect est la transmission différé de couches d'un même segment, ce qui est facilement faisable au niveau de l'implémentation des requêtes HTTP comme le montre l'architecture de la Fig. 7.12.

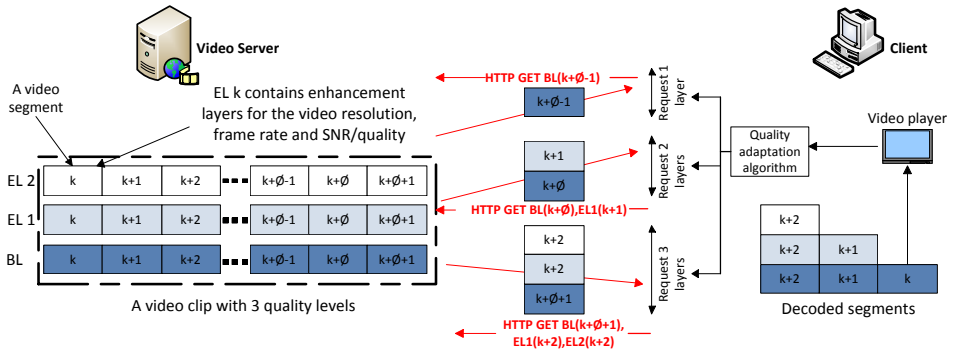

FIGURE 7.12: L'architecture BSC dans DASH

Chapitre 4
Analyse de performance de BSC

Nous allons dans ce chapitre analyser les performances du système BSC. Pour cela on va considérer la qualité d'expérience des utilisateurs. Celui-ci étant déterminé par les métriques suivantes: le délai initial, le taux de blocage de la vidéo, la qualité moyenne de la vidéo et la variabilité temporelle de la qualité. Il y a également l'impact de certains facteurs externes qui sont considérés dans les études empiriques de la qualité d'expérience. Dans notre analyse, nous calculons la probabilité de blocage et la probabilité de la fonction génératrice du nombre de blocages à travers le théorème du scrutin. Ensuite nous calculons le délai initial et la qualité moyenne de la vidéo en utilisant la distribution quasi-stationnaire des chaines de Markov plus précisément les processus de morts naissances qui seront utilisés pour modéliser le nombre de trames vidéo dans le tampon du lecteur. L'objectif final étant l'évaluation du système BSC, nous

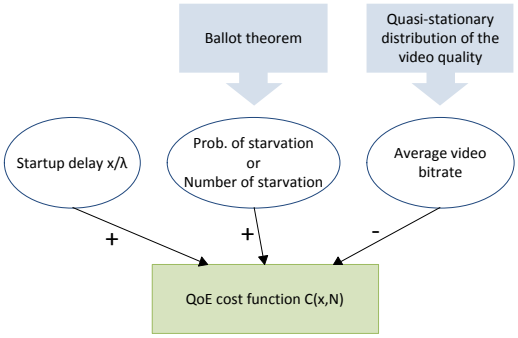

FIGURE 7.13: Le framework analytique pour l'évaluation du BSC

allons définir une fonction de coût de la qualité d'expérience qui prendra en compte les métriques définies ci-dessus comme le montre la Fig. 7.13.

La probabilité de blocage

Ici on va distinguer deux: le cas où $\phi \leq x$ et le cas où $\phi > x$ où ϕ est le décalage du système BSC et x est le nombre de trames vidéo stockées dans le tampon du lecteur avant de commencer la lecture de la vidéo. En utilisant le théorème du scrutin, on calculer la probabilité de blocage. Lorsque $\phi \leq x$, la probabilité de blocage est donnée par:

$$P_s^{<}(N,\phi,x) = \sum_{k=x+\phi-1}^{N-1} \frac{x+\phi-1}{2k-x-\phi+1} \binom{2k-x-\phi+1}{k-x-\phi+1} \cdot p^{k-x-\phi+1} q^k \tag{7.2}$$

Lorsque $\phi > x$, la probabilité de blocage est donnée par:

$$P_s^{>}(N,\phi,x) = P_{s1} + (1-P_{s1}) \cdot P_{s2} \tag{7.3}$$

où

$$P_{s1} = \sum_{k=x}^{\phi-2} \frac{x}{2k-x} \binom{2k-x}{k-x} \cdot p^{k-x} \cdot q^k \tag{7.4}$$

et

$$P_{s2} = \sum_{k=2\phi-2}^{N-1} \frac{x+\phi-1}{2k-x-\phi+1} \binom{2k-x-\phi+1}{k-x-\phi+1} \cdot p^{k-x-\phi+1} \cdot q^k \tag{7.5}$$

En utilisant la probabilité de blocage on calcule la probabilité de la fonction génératrice des événements de blocage.

La qualité moyenne de la vidéo

La qualité moyenne de la vidéo du système sans BSC est donnée par la qualité de la vidéo car celle ci ne varie pas. Par contre dans le cas du système BSC, la qualité de la vidéo varie entre la qualité de basse (qualité de la couche basique) et la qualité haute (qualité de la couche basique plus la couche d'amélioration). Pour calculer cette qualité moyenne, on va calculer le temps passé dans chaque qualité. Pour ce faire, on va modéliser le nombre de trames de couche basse et de couche haute dans le buffer comme le montre la Fig. 7.14. C'est une chaine de Markov, plus exactement un processus de

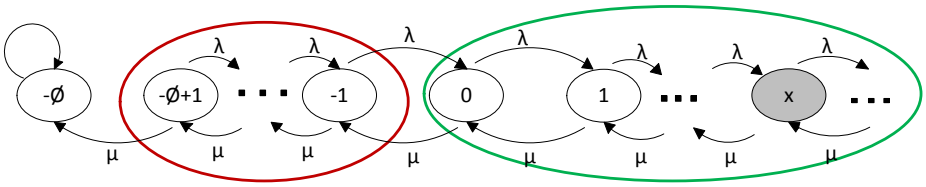

FIGURE 7.14: Processus de mort naissance modélisant le nombre de trames vidéo dans le buffer.

mort naissance homogène. Les transitions sont les taux d'arrivée et de départ des trames vidéo. L'état i correspond au nombre de trames dans le buffer. Lorsque $i \geq 0$, l'état i représente le nombre de trames de qualité haute dans le buffer. Lorsque $i < 0$, l'état i signifie qu'il y a $i + \phi$ trames dans le buffer. Pour calculer le temps passé dans chaque qualité, il nous faut calculer le temps passé dans chaque état i avant d'atteindre l'état absorbant $-\phi$: $E[S_i]$. On calcule la probabilité quasi-stationnaire de l'état i, q_i en remplaçant la chaine de Markov précédente par celle de la Fig. 7.15 où on supprime l'état absorbant et on ajoute une transition pour recommencer la lecture après un blocage. La probabilité q_i devient la probabilité du régime stationnaire $\mathbf{q}$ de la nouvelle chaine

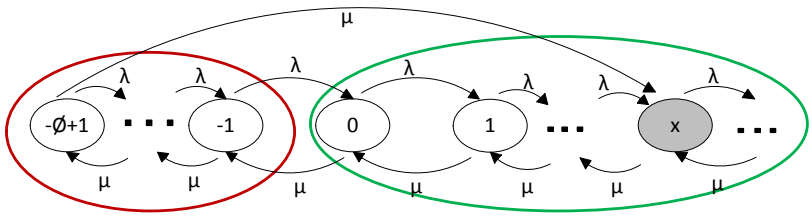

FIGURE 7.15: Processus de mort naissance modélisant le nombre de trames vidéo dans le buffer.

de Markov. **q** est donné en résolvant les équations de balance de la chaine de Markov.

$$\begin{cases} \mu q_{-\phi+2} & = & (\lambda+\mu)q_{-\phi+1} \\ \lambda q_{i-1} + \mu q_{i+1} & = & (\lambda+\mu)q_i, i \neq x+1 \\ \lambda q_{x-1} + \mu(q_{x+1} + q_{-\phi+1}) & = & (\lambda+\mu)q_x \\ \sum_{i=-\phi+1}^{\infty} q_i = 1 \end{cases} \tag{7.6}$$

$$\begin{aligned} q_i &= q_{-\phi+1}\frac{1-\rho^{i+\phi}}{1-\rho}, \quad i = -\phi+1, -\phi+2, .., x \\ q_i &= q_{-\phi+1}\frac{1-\rho^{x+\phi}}{1-\rho}\rho^{i-x}, \quad i = x+1, x+2, ... \end{aligned}$$

On trouve

$$\begin{cases} q_{-\phi+1} & = & \frac{1-\rho}{\phi+x} \\ q_i & = & \frac{(1-\rho^{i+\phi})}{(\phi+x)}, \quad i = -\phi+1, .., x \\ q_i & = & \frac{(1-\rho^{x+\phi})}{((\phi+x)}\rho^{i-x}, \quad i = x+1, x+2, ... \end{cases} \tag{7.7}$$

Le temps moyen passé dans l'état i est donné par

$$E[S_i] = q_i E[\tau_x] \tag{7.8}$$

où $E[\tau_x]$ est le temps total passé dans la chaine avant l'état absorbant en partant de l'état initial x. $E[\tau_x]$ est donné par [98]

$$E[\tau_x] = \frac{x+\phi}{\mu-\lambda} \tag{7.9}$$

Ce qui donne

$$E[S_i] = q_i\frac{x+\phi}{\mu-\lambda} \tag{7.10}$$

Soit $T_{\mathcal{L}}$ et $T_{\mathcal{H}}$ le temps passé dans la qualité basse et la qualité haute respectivement. On a

$$T_{\mathcal{L}} = \sum_{i=-\phi+1}^{-1} E[S_i] \text{ and } T_{\mathcal{H}} = \sum_{i=0}^{\infty} E[S_i] \tag{7.11}$$

La qualité moyenne du système BSC est donné par

$$b_{avg} = \frac{T_{\mathcal{L}} b_{\mathcal{L}} + b_{\mathcal{H}} T_{\mathcal{H}}}{T_{\mathcal{L}} + T_{\mathcal{H}}} \tag{7.12}$$

où $b_{\mathcal{L}}$ et $b_{\mathcal{H}}$ sont le débit binaire de la qualité basse et de la qualité haute respectivement.

Le délai initial

Le temps d'attente initial est donné par $\frac{x}{\lambda}$. Par contre après un blocage, le temps d'attente est donné par $\frac{x+\phi-1}{\lambda}$.

La fonction de coût de la QoE

Pour évaluer le système BSC, on va définir une fonction de coût de la qualité d'expérience $c(x, N)$ en prenant en compte les métriques calculées précédemment.

$$C(x, N) = \gamma_1 \cdot E[T_x] + \gamma_2 . P_s - \gamma_3 \cdot \sum_{i=1}^{r} w_i T_i \tag{7.13}$$

Dans cette fonction, $E[T_x]$ est le temps d'attente initial, P_s est la probabilité de blocage et $\sum_{i=1}^{r} w_i T_i$ représente la qualité moyenne de la vidéo avec w_i le rapport débit binaire sur la durée de la vidéo et T_i le temps passé dans la qualité i. Les paramètres γ_1, γ_2 et γ_3 permettent d'ajuster les métriques de la QoE en fonction de la préférence des utilisateurs.

Les simulations et les résultats numériques montrent que le système BSC permet de réduire la probabilité de blocage du lecteur vidéo tout en améliorant la qualité de la vidéo. Dans le prochain chapitre nous étudierons l'adaptation de la qualité de la vidéo dans le système BSC.

Chapitre 5
Algorithmes d'adaptation de BSC

Dans ce chapitre nous proposons des algorithmes d'adaptation pour le système BSC. La vidéo est divisée en K segments. Chaque segment de longueur T_s secondes est encodé en plusieurs qualités correspondantes à plusieurs couches: une couche basique et plusieurs couches d'amélioration. Dans les systèmes DASH utilisant le codec SVC, la transmission des couches d'un segment se fait simultanément. Par contre dans le système BSC, la transmission des couches d'un segment se fait deux fois: un segment de couche basse contenant la couche basique et probablement quelques couches d'amélioration, et un segment de couche haute contenant uniquement des couches d'amélioration. Le segment de couche haute est transmis après le segment de couche basse $\phi - 1$ segments plus tard. La superposition d'un segment de couche basse et d'un segment de couche

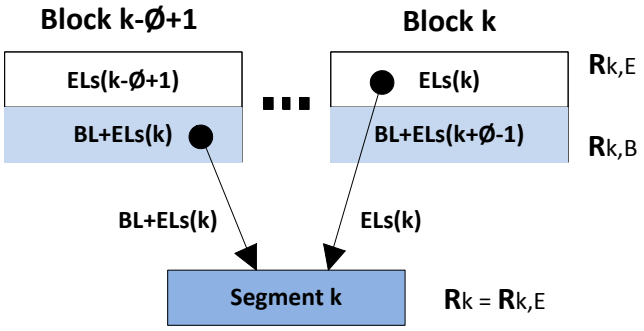

FIGURE 7.16: La transmission du segment de couche basse et du segment de couche haute et le décodage pour avoir le segment correspondant.

haute est appelé bloc. Dans la Fig. 7.16, le segment de couche basse du segment k est transmis dans le bloc $k - \phi + 1$. A cet instant le segment peut être décodé pour avoir le segment k avec la qualité correspondante $R_{k,B}$. Puis plus tard, dans le bloc k, le segment de couche haute du segment k est transmis. A l'aide de ces couches d'amélioration on accroit la qualité du segment k à $R_{k,E}$. Soit $\mathcal{R}$ l'ensemble des qualités dans le système classique DASH utilisant SVC. L'objectif dans ce chapitre est de proposer des algorithmes pour sélectionner les qualités des segments de couche basse et de couche haute $(R_{k,B}, R_{k,E}) \in \mathcal{R}$.

Les algorithmes d'adaptation proposés

Le but de l'adaptation de la qualité de la vidéo à la capacité du réseau est de maximiser la qualité d'expérience des utilisateurs de vidéo streaming. Ce qui revient à trouver un compromis entre les métriques suivantes: le temps d'attente initial, les risques d'interruption de la vidéo, la qualité moyenne de la vidéo et le nombre moyen de variations de la qualité. L'objectif est donc de proposer des algorithmes d'adaptation de la qualité qui vont tenir compte de ces métriques. La Fig. 7.17 montre le diagramme de flux des algorithmes proposés. Ils tiennent compte de deux choses: le débit estimé du réseau et la taille de données vidéo stockés dans le buffer du lecteur. Le terme TB-BSC fait référence aux algorithmes utilisant le débit estimé tandis que BB-BSC fera référence

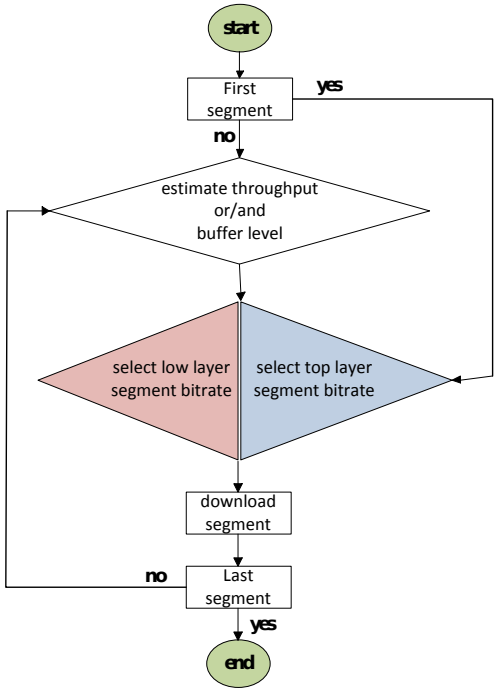

FIGURE 7.17: Diagramme de flux pour les algorithmes d'adaptation de BSC.

aux algorithmes utilisant la taille du buffer. Le BSC est capable d'adapter n'importe quel algorithme de la littérature DASH, on va donc considérer les algorithmes suivants. Le premier sera l'algorithme TB-SVC, il utilise le débit estimé pour décider de la qualité du segment à télécharger. L'algorithme BBA-0 utilise trois seuils de buffer pour décider de la qualité du segment à télécharger en fonction de la taille des données dans le buffer. Enfin l'algorithme BBA-1 va utiliser une fonction d'ajustement.

L'algorithme TB-BSC

Soit k le bloc à télécharger. $\hat{A}_t$ et $\mathcal{B}_k$ représentent, respectivement, le débit estimé et la taille des données dans le buffer en secondes. R_{min} et R_{max} sont, respectivement, le plus petit et le plus grand débit binaire dans l'ensemble $\mathcal{R}$. On va distinguer deux cas: le cas $k < \phi$ et le cas $k \geq \phi$.

Lorsque $k < \phi$, le segment de couche basse contient uniquement la couche basique, ce qui correspond au débit binaire R_{min}. L'adaptation sur le segment de couche haute est faite de telle sorte que $R_k + R_{min} \leq \hat{A}_t$. On assignant R_{min} au segment de couche basse, on maximise tout de suite la qualité du segment de couche haute et on impacte moins le temps d'attente initial.

Lorsque $k \geq \phi$, on va estimer le débit et sélectionner $R_{k,B}$ et $R_{k,E}$ en fonction du débit estimé. Soit $\phi_t = \phi.T_s$ la longueur du offset en secondes. Lorsque la taille du buffer $\mathcal{B}_k$ est inférieure à ϕ_t, on ne transmet aucun segment de couche haute car soit les segments de couche basse correspondants ont déjà été joué, soit ils seront joué avant la fin du téléchargement des segments de couche haute. Dans ce cas, l'adaptation est équivalente au cas DASH utilisant SVC. Par contre lorsque $\mathcal{B}_k > \phi_t$, on peut transmettre des segments de couche haute. On choisit le débit binaire juste en dessous du débit

estimé pour le segment de couche basse et le prochain débit binaire pour le segment de couche haute.

L'algorithme BB-BSC

On appelle BB-BSC-0, l'algorithme de BSC qui utilise BBA-0. Dans ce cas, des seuils de buffer sont fixés et en fonction du remplissage du buffer on croit, décroit ou garde le même débit binaire pour le prochain segment à télécharger. L'algorithme est similaire au précédent en ce qui concerne $\mathcal{B}_k$ et ϕ_t.

BB-BSC-1 est l'algorithme de BSC qui utilise l'algorithme BBA-1. On définit deux fonctions d'ajustement F_2 et F_2. F_1 est utilisé pour sélectionner la qualité du segment de couche basse tandis que F_2 est utilisé pour sélectionner la qualité du segment de couche haute comme le montre la Fig. 7.18. L'objectif de la fonction F_2 n'est pas uniquement

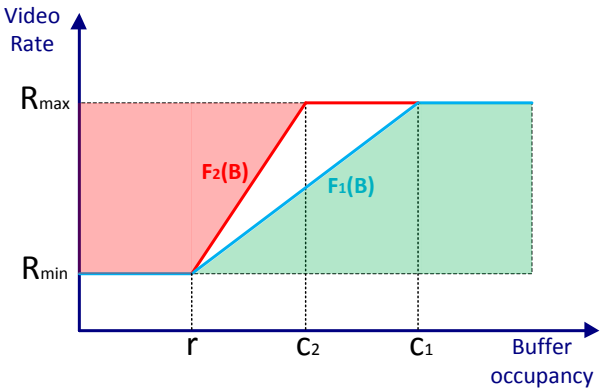

FIGURE 7.18: Les fonctions d'ajustement pour la sélection des débits binaires du segment de couche basse et du segment de couche haute.

d'améliorer la qualité de la vidéo mais aussi de réduire le nombre de variations de qualité. Pour cela, on applique cette fonction simultanément sur un ensemble de $\phi - 1$ segments comme le montre la Fig. 7.19.

Exemples de résultats numériques

Nous avons évaluer le système BSC et comparer les algorithmes proposés avec ceux du standard DASH. Nous avons utilisé deux types de capacité de réseaux: le premier est généré avec matlab tandis que le deuxième représente des traces de débit de réseaux HSDPA. L'utilisateur télécharge la vidéo segment par segment en envoyant des requêtes HTTP au serveur. Les segments téléchargés sont stockés dans le buffer du lecture pour être lus.

On a d'abord comparé l'algorithme TB-BSC à son équivalent DASH TB-SVC dans la Fig. 7.20. L'algorithme TB-BSC a un meilleur débit binaire que TB-SVC, 2040Kbps contre 1382Kbps. De plus il gère aussi bien la variation de la qualité au cours de la session vidéo: 10 pour TB-BSC contre 9 pour TB-SVC.

Dans la Fig. 7.21, on compare l'algorithme BB-BSC-1 contre l'algorithme de DASH BBA-1. Le débit binaire moyen est de 689Kbps pour BB-BSC-1 contre 660Kbps pour BBA-1. Même si le débit n'est pas fortement amélioré (celui dépend de la capacité du réseau), on constate une très grande amélioration au niveau de la variation de qualité: 10 pour BB-BSC-1 contre 22 pour BBA-1. On a comparé tous les algorithmes

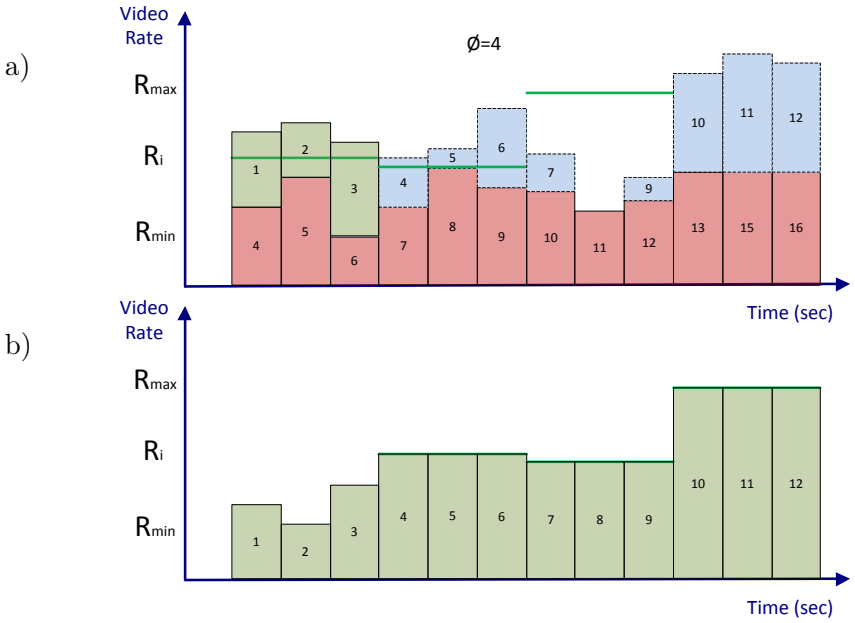

FIGURE 7.19: a) Exemple d'application de la fonction F_2 b) Le rendu au niveau du décodeur; $\phi = 4$.

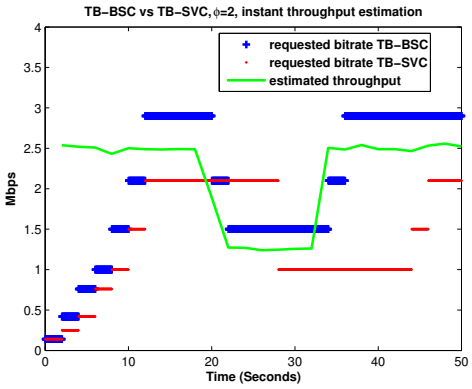

FIGURE 7.20: débit binaire des segments pour TB-BSC et TB-SVC

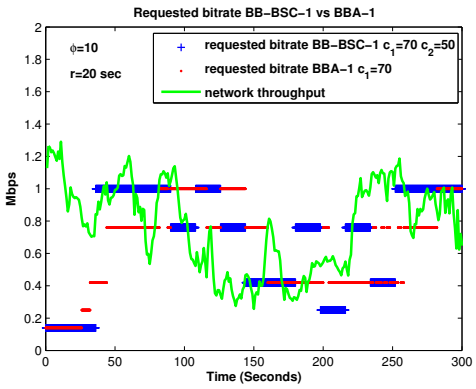

FIGURE 7.21: débit binaire des segments pour BB-BSC-1 et BBA-1, c_1 et BBA-1 est 70 sec

dans le tableau 7.1: BBA-0 pour l'algorithme de DASH utilisant les seuils du buffer, BBA-0 pour l'algorithme de DASH utilisant la fonction d'ajustement, TB-BSC pour l'algorithme de BSC utilisant le débit estimé pour la selection du débit binaire des segments, enfin BB-BSC-1 est l'algorithme de BSC utilsant deux fonctions d'ajustement. Les algorithmes BSC ont un meilleur débit binaire par rapport aux algorithmes de DASH. De plus, on réduit énormément le nombre de variations de qualité lorsqu'on utilise la taille des données dans le buffer pour décider du débit binaire des segments dans le cas de BSC. Le nombre de blocages moyen est 0 dans les quatre cas car on voit sur les figures précédentes que le débit du réseau est toujours supérieur à la plus petite qualité de la vidéo. On a montré dans ce chapitre l'intégration du BSC dans le

	Qualité moyenne (Kbps)	Variance de la qualité	Nombre de variations de la qualité	Nombre de blocages
BBA-0	581	281e9	13.61	0
BBA-1	634	809e9	21.38	0
TB-BSC	**698**	950e9	58	0
BB-BSC-1	**687**	**116e9**	**10**	0

TABLE 7.1: métriques de la QoE: qualité moyenne, variabilité de la qualité, nombre de variations de la qualité et nombre de blocages.

standard DASH. BSC peut utiliser n'importe quel algorithme de DASH et l'adaptant à sa manière de faire l'adaptation, c'est à dire transmettre le segment avec une qualité donnée (en fonction de la capacité du réseau) puis améliorer la qualité de ce segment plus tard en transmettant des couches d'amélioration SVC. Dans le chapitre suivant on va étudier le caching de la vidéo dans le cas du streaming adaptatif utilisant le codec SVC.

Chapitre 6
Caching dans les systèmes de streaming adaptatif

Le réseau cœur est devenu ces dernières années, le goulot, le point de congestion de la transmission. Ceci est dû en partie à l'explosion du trafic de vidéo streaming. Le caching est une technique, déjà utilisé dans d'autres domaines, qui permet de déployer des serveurs de petite capacité dans la partie radio du réseau afin de stocker certaines données. Les algorithmes de caching existant comme LRU et LFU échouent à optimiser le contenu des caches car le paradigme de la vidéo a changé avec l'avènement de la vidéo streaming adaptatif. En effet, un contenu vidéo n'est plus représenté par un fichier unique mais par plusieurs fichiers, chacun correspondant à une qualité appelé représentations. La question qui se pose alors est quelle représentation faut-il stocker dans le cache si besoin? Nous allons dans cette partie proposer plusieurs algorithmes de caching qui vont optimiser le contenu des caches afin de minimiser le trafic généré sur le réseau cœur.

System

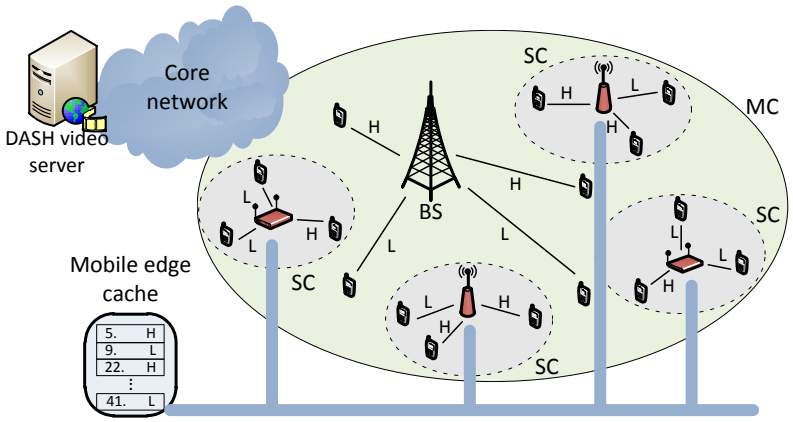

FIGURE 7.22: Architecture du système: les utilisateurs sont connectés au serveur distant à travers l'interface radio et un proxy qui intercepte les requêtes en utilisant des copies pour mettre en cache les vidéos.

On considère un ensemble de $\mathcal{U} = \{1, 2, ..., U\}$ utilisateurs dans une macro cellule avec une station de base. Un cache de capacité M octets est installé au niveau de la station de base. Un ensemble $\mathcal{I} = \{1, 2, ..., I\}$ de vidéos sont stockées dans le serveur du fournisseur de contenu derrière le réseau cœur. Chaque vidéo est disponible en L qualités distinctes. Les fichiers étant encodés en utilisant le codec SVC. R_1, R_2,..., R_L sont les débits binaires correspondants aux différentes qualités. La durée de lecture de la vidéo i de qualité R_l est T_i, ce qui correspond à une taille de $R_l.T_i$.

Soit ω_i le taux de demande totale pour la vidéo i. On suppose que la capacité du canal de chaque UE prend sa valeur dans l'ensemble $\mathcal{C} = \{C_1, C_2, ..., C_L\}$. On suppose que lorsque le canal de l'UE a une capacité $C_u(t) = C_k$ pour $1 \leq k \leq L$, il demande la vidéo avec la qualité R_k. Dans le cas de SVC, lorsqu'une requête arrive pour une vidéo avec une qualité strictement supérieure (R_k) à la qualité se trouvant dans le cache (R_l),

il suffit juste de télécharger du serveur, les couches manquantes pour satisfaire la qualité demandée. Le trafic généré à travers le serveur s'écrit $[R_k - R_l]^+$, où $[b]^+ = \max\{0, b\}$.

Solution optimale

La solution optimale correspond à la solution de caching qui va minimiser le trafic généré au travers du serveur distant. Soit $x_{i,l}$ la variable correspondante à l'évènement le contenu i est stocké dans le cache avec la qualité R_l où $\sum_{l=1}^{L} x_{i,l} = 1$. On suppose que la probabilité stationnaire ergodique qu'un utilisateur a son canal dans l'état R_k est $\pi_k = \mathbb{P}\{C_u(t) = R_k\}$.

Dans le cas le plus simple où chaque utilisateur a un canal statique et n_k est le nombre d'utilisateurs avec l'état du canal C_k, $k = 1, \ldots, L$, on a $\pi_k = n_k/U$. Le trafic moyen généré au travers du serveur distant s'écrit

$$
\begin{aligned}
\mathrm{E}[\tau_i] &= \omega_i T_i \sum_{k=1}^{L} \left[R_k - \sum_{l=0}^{L} R_l \, x_{i,l} \right]^+ \pi_k \\
&= \omega_i T_i \sum_{k=1}^{L} \Big(R_k - \sum_{l=0}^{k-1} R_l x_{i,l} - \sum_{l=k}^{L} R_l x_{i,l} \Big) \pi_k \\
&= \omega_i T_i \left[\sum_{k=1}^{L} \sum_{l=0}^{k} \Big(\frac{R_k}{k+1} - R_l \cdot x_{i,l} \Big) \pi_k - \sum_{k=1}^{L} R_k \pi_k \sum_{l=k}^{L} x_{i,l} \right] \\
&= \omega_i T_i \left[\sum_{l=1}^{L} \sum_{k=l}^{L} \Big(\frac{R_k}{k+1} - R_l \cdot x_{i,l} \Big) \pi_k + \sum_{k=1}^{L} \frac{R_k}{k+1} \pi_k \right. \\
&\quad \left. - \sum_{l=1}^{L} \sum_{k=1}^{l} R_k \pi_k x_{i,l} \right] \\
&= \omega_i T_i \Big(\sum_{l=1}^{L} \sum_{k=l}^{L} \frac{R_k}{k+1} \pi_k + \sum_{k=1}^{L} \frac{R_k}{k+1} \pi_k \Big) \\
&\quad - \omega_i T_i \sum_{l=1}^{L} \Big(R_l \sum_{k=l}^{L} \pi_k + \sum_{k=1}^{l-1} R_k \pi_k \Big) x_{i,l}
\end{aligned}
$$

On peut observer que le premier terme ne dépend pas de la politique de caching. Alors, le but est de maximiser la fonction objective

$$
\Lambda = \sum_{i=1}^{I} \Lambda_i = \sum_{i=1}^{I} \omega_i T_i \sum_{l=1}^{L} E_l \, R_l \, x_{i,l} \tag{7.14}
$$

où

$$
E_l = R_l \sum_{k=l}^{L} \pi_k + \sum_{k=1}^{l-1} R_k \pi_k
$$

L'équation 7.14 représente le contenu qu'on est en mesure de servir directement aux utilisateurs à partir du cache. La solution est donnée donc en formulant le problème de programmation linéaire suivant

Problem 2. *Déterminer la stratégie d'allocation statique qui maximise*

$$\textit{maximiser} \quad \sum_{i=1}^{I} \omega_i T_i \sum_{l=0}^{L} E_l \, R_l \, x_{il} \tag{7.15}$$

$$\textit{en fonction de} \quad \sum_{i=1}^{I} \sum_{l=1}^{L} T_i \, R_l \, x_{il} \leq M, \tag{7.16}$$

$$\sum_{l=0}^{L} x_{il} = 1, \quad i = 1, 2, ..., I, \tag{7.17}$$

$$x_{il} \in \{0, 1\}, i = 1, 2, ..., I, \; l = 1, 2, ..., L \tag{7.18}$$

où M est la capacité du cache, x_{il} signifie que le contenu i est stocké avec la qualité R_l. La condition 7.17 signifie qu'une seule qualité du contenu i est stocké dans le cache à la fois.

Algorithmes proposés

On a d'abord adapté les algorithmes existants dans le cas du codage SVC. Il faut noter que lorsqu'on utilise un codage uni-couche, par exemple AVC, chaque représentation appelée version est indépendante de l'autre, on peut donc les considérer comme des contenus différents et appliquer les algorithmes comme LRU, LFU, etc. Dans ce cas, le cache n'est pas optimisé car on peut se retrouver avec plusieurs versions d'un même contenu dans le cache. La question alors est quelle qualité stockée dans le cache? On a proposé deux algorithmes pour décider de la qualité à stocker dans le cache.

MRQ: C'est un algorithme qui va stocker la dernière qualité demandée dans le cache. Ainsi, si une qualité supérieure se trouve dans le cache, il va supprimer les couches d'amélioration de trop. Par contre, lorsqu'une qualité inférieure est dans le cache, il va télécharger les couches d'amélioration nécessaires du serveur.

MFQ: C'est un algorithme qui va stocker dans le cache, la qualité la plus fréquemment demandée. Ces algorithmes s'occupent de la qualité de la vidéo. En ce qui concerne la vidéo en tant que entité unique, les algorithmes existants peuvent encore utiliser.

En plus dans le cas du codage SVC, lorsqu'on veut faire de la place dans le cache, on peut commencer par supprimer les couches d'amélioration d'un contenu au lieu de tout le fichier. Ceci permet d'augmenter le hit ratio.

L'algorithme CQM: C'est un algorithme qui va reproduire la distribution de l'état du réseau dans le cache. Les algorithmes existants sont d'abord utilisés pour ranger les vidéos par ordre de popularité dans le cache. Ensuite on calcule à partir des requêtes des utilisateurs le nombre de vidéos pour chaque qualité devant de trouver dans le cache. La Fig. 7.23 montre un exemple d'application de l'algorithme CQM. Les vidéos les plus populaire (plus grand nombre de requêtes) sont stockées dans le cache en suivant la distribution des requêtes. Ici les quatre premières vidéos sont stockées avec la qualité de $720p$, les cinq suivantes avec la qualité de $480p$ et les deux dernières à être dans le cache avec une qualité de $360p$.

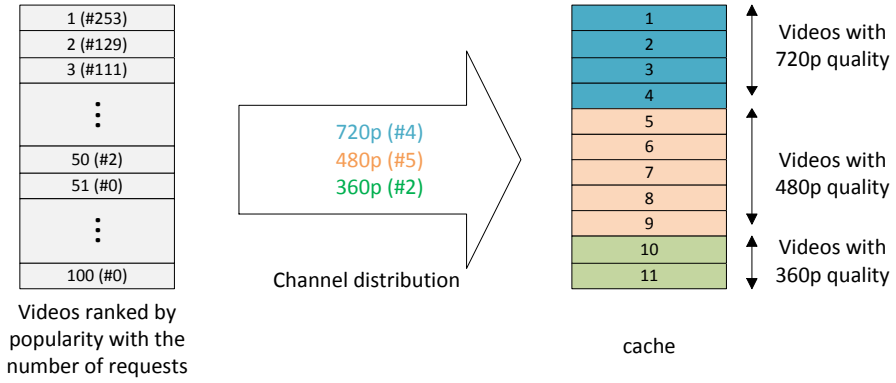

FIGURE 7.23: L'algorithme CQM

Résultats numériques

Nous avons simulé notre système avec 100 vidéos. Chacune étant disponible en cinq qualités, i.e., {250, 400, 750, 1000, 1200} avec une durée de 300 secondes. La popularité des vidéos suit la loi de Zipf's de paramètre $\alpha = 0.5$. On a utilisé une distribution uniforme pour les requêtes de qualités.

Nous avons tracé dans la Fig. 7.24 et 7.25, respectivement, le trafic généré à travers le serveur distant et le hit ratio pour les algorithmes décrits ci-dessus. La première

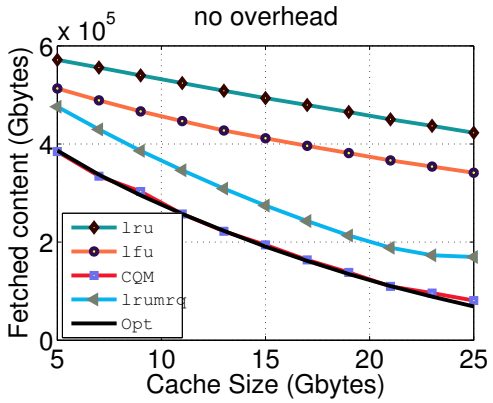

FIGURE 7.24: Comparaison entre les algorithmes proposés pour le trafic généré sur le serveur

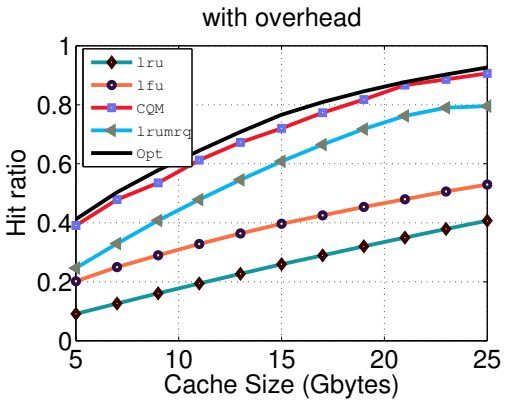

FIGURE 7.25: Comparaison entre les algorithmes proposés pour le hit ratio

constatation est que les algorithmes pour la qualité à savoir MRQ et MFQ associés aux algorithmes existants tels que LRU ou LFU sont plus performants que les algorithmes LRU et LFU seuls utilisant le codage AVC. L'algorithme CQM est celui qui s'approche le plus de la solution optimale de caching. Donc tenir compte de la distribution de la qualité des requêtes est primordial dans les systèmes de streaming adaptatif pour le caching. Le hit ratio dans la Fig. 7.25 confirme les resultats de la Fig. 7.24.

Chapitre 7
Conclusions et perspectives

L'amélioration des technologies de vidéo streaming est cruciale à cause de l'explosion du trafic vidéo ces dernières années. En effet ce trafic va représenter 75% du trafic total Internet d'ici 2020. De plus avec le réseau cœur qui devient de plus en plus le goulot de transmission, il est nécessaire d'utiliser des techniques comme le caching pour soulager les liens de ce réseau cœur. Dans cette thèse nous avons proposé deux solutions pour améliorer la transmission de la vidéo streaming dans les réseaux mobiles: le BSC qui est un nouveau schéma de transmission de la vidéo streaming basé sur le codec multi-couches SVC et de nouveaux algorithmes de caching tenant compte de la qualité de la vidéo demandée. Nous avons expliqué le fonctionnement du schéma BSC et son implémentation dans le standard DASH. Puis nous avons effectué une étude analytique en se basant sur la qualité d'expérience des utilisateurs pour évaluer les performances du système BSC. Nous avons ensuite proposé des algorithmes d'adaptation qui ont permis d'améliorer la qualité de la vidéo de 19% et surtout de diminuer le nombre de variations de qualité soit une amélioration allant jusqu'à 36%.

Enfin nous avons proposé des algorithmes de caching dans le contexte de la vidéo streaming adaptatif pour tenir compte de la qualité des vidéos demandées.

Comme perspectives, nous pourrons nous intéresser au comportement du système BSC lorsque plusieurs utilisateurs compétissent pour la même bande passante ou utiliser des techniques d'ordonnancement au niveau de la station de base pour améliorer la qualité de la vidéo. Nous pouvons aussi étudier des modèles analytiques pour le problème d'adaptation de la qualité.

Pour le caching on pourra étudier le problème de caches multiple. Dans ce cas, on parlera de politique de caching et de politique de replication. C'est à dire dans quels caches faut-il stocker la vidéo. On pourra aussi étudier la caching en se plaçant côté utilisateur donc trouver une solution de caching optimale qui maximise la qualité d'experience.