QoE-Based Scheduling Algorithms for Adaptive HTTP Video Delivery in LTE

Frederico Freire Rodrigues

Thesis to obtain the Master of Science Degree in

Electrotecnical and Computer Engineering

Supervisors

Prof. António José Castelo Branco Rodrigues

Prof. Maria Paula dos Santos Queluz Rodrigues

Dr. Ivo Luís de la Cerda Garcia e Sousa

Examination Committee

Chairperson: Prof. José Eduardo Charters Ribeiro da Cunha Sanguino

Supervisor: Prof. António José Castelo Branco Rodrigues

Member of the Committee: Prof. Pedro Manuel de Almeida Carvalho Vieira

November 2017

Acknowledgements

I would like to thank my supervisors, Prof. Antonio Rodrigues, Prof. Maria Paula Rodrigues and Dr. Ivo Sousa for the invaluable guidance, patience and support given throughout the development of this work.

A heartfelt thank you also goes to my family and friends, without whom none of this would have been possible.

Resumo

Nos últimos anos, o consumo de conteúdo multimédia aumentou exponencialmente, em particular o de vídeo. Hoje em dia, os operadores móveis têm dois desafios. Por um lado, necessitam de satisfazer as altas expectativas que os utilizadores têm nos serviços utilizados. Por outro lado, a capacidade da rede móvel não consegue ser aumentada tão rapidamente como a necessidade o exige. Uma solução possível passa pelo desenvolvimento de estratégias de alocação de recursos inteligentes que aloquem os recursos da rede de forma eficiente e que satisfaçam o maior número de clientes, proporcionando-lhes uma elevada qualidade de experiência (QoE).

Motivado pelos algoritmos de alocação de recursos que se podem encontrar na literatura, nesta tese apresenta-se o seu estado da arte, analisando as debilidades dos mesmos, bem como se propõe uma nova e eficaz solução, que pode ser usada numa rede *Long Term Evolution* (LTE). É proposto um algoritmo de alocação de recursos, *Maximum Buffer Filling* (MBF), que permite aumentar o número de utilizadores móveis satisfeitos que fazem *streaming* de vídeo. Para isso, o MBF faz uso do estado do buffer dos utilizadores, que se trata de uma métrica de QoE reportada pelos clientes que implementam a especificação *Dynamic Adaptive Streaming over HTTP* (DASH), estandardizada pela MPEG e também conhecida por MPEG-DASH. Considera também o estado do canal de rádio que os utilizadores reportam. O MBF aloca recursos de acordo com o nível do buffer dos utilizadores e com a quantidade de segundos que conseguirão fazer download. O algoritmo pode operar em dois modos (o modo 1 e o modo 2), de acordo com a objetivo do operador: aumentar o número de utilizadores satisfeitos ou minimizar o número de utilizadores insatisfeitos. Esta tese compara a capacidade suportada pelo MBF com as capacidades dos algoritmos de alocação *Round Robin* (RR), *Blind Equal Throughput* (BET), *Proportional Fair* (PF) e *Proportional Fair with Barriers for Frames* (PFBF), quanto ao número de utilizadores com boa ou excelente QoE que fazem *streaming* de vídeo. Também é analisado o número de utilizadores não satisfeitos (com má ou pobre QoE). Se se procurar garantir que 90 % dos utilizadores têm uma boa ou excelente QoE, independentemente do modo de operação do MBF, este suporta claramente mais utilizadores conectados do que o RR, BET, PF e PFBF. Em comparação com o RR, o MBF suporta mais 15% e 12% de utilizadores conectados, operando no modo 1 e modo 2, respetivamente. Para além disto, o MBF permite ter menos utilizadores com uma má ou pobre QoE do que o RR, o PF e PFBF, independentemente do modo de operação.

Palavras-chave: streaming móvel de vídeo, DASH, algoritmos de alocação de recursos, qualidade de experiência, throughput, buffer.

Abstract

In the last years, the multimedia content consumed by mobile users has exponentially increased, particularly video content. Today, mobile operators have two challenges. On the one hand, they need to satisfy the high expectations that clients have on the delivered quality of the services. On the other hand, mobile network capacity cannot be increased as fast as the demand growth. A possible solution is the development of intelligent schedulers that allocate resources very efficiently and satisfy the maximum possible number of clients, providing them with a good Quality of Experience (QoE).

Motivated by the scheduling algorithms that can be found in the literature, this thesis presents an overview of the state-of-the-art – notably to understand the current weaknesses – and proposes a new and effective solution that can be used in a Long Term evolution (LTE) network. It proposes a scheduler, Maximum Buffer Filling (MBF), that increases the number of users who are satisfied with their video streaming session. To do so, it makes use of the user's current buffer level which is a QoE metric reported by the clients which implement the Dynamic Adaptive Streaming over HTTP specification (DASH), standardized by MPEG and also known as MPEG-DASH. The reported radio channel status and video segment requests sent by the users to the base station are also considered in the scheduling process. MBF allocates resources according to the current buffer level of the users and their achievable buffer filling. It can operate in two modes (mode 1 and mode 2), according to operator's intention: to maximize the number of satisfied users or to minimize the number of non-satisfied ones. This thesis assesses the capacity provided by MBF and compares it with the ones provided by Round Robin(RR), Blind Equal Throughput (BET), Proportional Fair (PF) and Proportional Fair with Barriers for Frames (PFBF) schedulers, regarding the maximum number of satisfied users who have a good or excellent QoE streaming video. It is also analyzed the number of non-satisfied users (with a poor or bad QoE). To ensure that 90% of the users have a good or excellent QoE, MBF, regardless the mode in which it operates, clearly supports a higher number of users streaming video than RR, BET, PF and PFBF. With respect to RR, it supports more 15% and 12% of connected users, when operating in mode 1 and mode 2, respectively. Furthermore, MBF leads also to a smaller number of users with a bad or poor QoE than RR, PF and PFBF, regardless the mode in which it operates.

Keywords: mobile video streaming, DASH, scheduling algorithms, quality of experience, throughput, buffer.

Contents

Acknowledgements iii

Resumo v

Abstract vii

List of figures xi

List of tables xiii

List of acronyms xv

1 Introduction **1**

1.1 Context and motivation 1

1.2 Objectives 2

1.3 Main contributions 3

1.4 Thesis outline 3

2 DASH overview **5**

2.1 Introduction 5

2.2 Adaptive HTTP streaming 5

2.3 Scope of MPEG-DASH 7

2.4 DASH client model 9

2.5 DASH data model 10

2.6 Protocols 12

2.7 Media content 13

2.7.1 Format 13

2.7.2 Segmentation 13

2.7.3 Reproduction 14

2.8 DASH profiles 14

3 QoE overview **17**

3.1 Introduction 17

3.2 Factors influencing QoE 17

3.3 QoE measurement methods 18

3.3.1 Subjective assessment 18

3.3.2 Objective assessment 18

3.4 QoE in DASH 20

3.4.1 QoE triggering and reporting 20

3.4.2 QoE metrics 21

3.4.3 Quality metadata 23

4 Scheduling algorithms **25**

4.1 Introduction 25

4.2 Design aspects 27

4.3 QoE-unaware schedulers 28

4.3.1 Channel-unaware strategies 28

4.3.2 Channel-aware / QoS-unaware strategies 30

4.3.3 Channel-aware / QoS-aware strategies 32

4.4 QoE-aware schedulers 33

4.4.1 Passive end-user device strategies 34

4.4.2 Active end-user device / passive user strategies 36

4.4.3 Active end-user device / active user strategies 37

4.5 Proposed QoE-aware scheduling algorithm 38

4.5.1 Algorithm development aspects 38

4.5.2 A new algorithm: Maximum Buffer Filling 42

5 Simulator implementation **45**

5.1 Introduction 45

5.2 User CQI, throughput and buffer 45

5.3 QoE adaptation algorithm 47

5.4 QoE model 53

5.5 Implemented scheduling algorithms 54

6 Simulations and results **59**

6.1 Introduction 59

6.2 Monte Carlo method 59

6.3 Simulation parameters 60

6.4 Monte Carlo method demonstration 61

6.5 Algorithms comparison methodology 62

6.5.1 Analysis of the number of satisfied users 65

6.5.2 Analysis of the percentage of users with global QoE < 2 71

6.6 Results summary 72

7 Conclusions **75**

References **77**

A Appendix **81**

List of figures

Figure 1.1 - Three levels of QoS proposed in [3]. 1
Figure 2.1 - Evolution of streaming toward adaptive HTTP streaming. 6
Figure 2.2 - Workflow of adaptive HTTP Streaming. 7
Figure 2.3 - MPEG-DASH system description. 8
Figure 2.4 - DASH client model [9]. 9
Figure 2.5 - Client streaming process. 9
Figure 2.6 - DASH data model [8]. 10
Figure 2.7 - DASH data model representation. 12
Figure 2.8 - Overview of the protocol stack [9]. 12
Figure 2.9 - MPEG-DASH profiles as defined in ISO/IEC 23009 [8]. 15
Figure 3.1 - Reporting framework for 3GPP DASH [15]. 20
Figure 4.1 - Time-frequency radio resources grid [20]. 25
Figure 4.2 - QoE-oriented scheduling strategies classification [27]. 34
Figure 4.3 - 1st Attempt - Percentage of satisfied users as a function of connected users streaming video. 40
Figure 4.4 - 2nd Attempt - Percentage of satisfied users as a function of connected users streaming video. 41
Figure 4.5 - Difference between the two MBF attempts regarding the number of users with global QoE < 2. 42
Figure 5.1 - User throughput for the step-down test. 49
Figure 5.2 - Estimated bandwidth and requested segment bitrate for the step-down test, achieved using the developed simulator. 49
Figure 5.3 - User buffer length for the step-down test, achieved using the developed simulator. 50
Figure 5.4 - Estimated bandwidth and requested segment bitrate for the step-down test, presented in [52]. 50
Figure 5.5 - User buffer length for the step-down test, presented in [52]. 51
Figure 5.6 - User throughput for the fluctuation test. 51
Figure 5.7 - Estimated bandwidth and requested segment bitrate for the step-down test, achieved using the developed simulator. 52
Figure 5.8 - Estimated bandwidth and requested segment bitrate for the step-down test, presented in [52]. 52
Figure 5.9 - User buffer length for the fluctuation test, achieved using the developed simulator. 52
Figure 5.10 - User buffer length for the fluctuation test, presented in [52]. 53
Figure 5.11 - CQIs reported by users along time for the RR test. 55
Figure 5.12 - Total number of allocated RB along time for the RR test. 55
Figure 5.13 - CQIs reported by users along time for the BET test. 55
Figure 5.14 - Users' average throughputs for the BET test. 56

Figure 5.15 - CQIs reported by users along time for the PF test. 56
Figure 5.16 - Users' average throughputs for the PF test. 57
Figure 5.17 - User metrics for the PF test. 57
Figure 5.18 - CDF of re-buffering percentage for PFBF and PF algorithms. 58
Figure 6.1 - Illustration of confidence intervals in the normalized normal distribution. 60
Figure 6.2 - Monte Carlo method demonstration using one single simulation and $Smin$ simulations. 61
Figure 6.3 - Influence of the parameter α [s] in the percentage of users with global QoE ≥ 3. 62
Figure 6.4 - Influence of the parameter α [s] in the percentage of users with global QoE ≥ 4. 63
Figure 6.5 - Influence of the parameter α [s] in the percentage of users with global QoE < 2. 63
Figure 6.6 - Percentage of satisfied users as a function of connected users streaming video (U = 3). 66
Figure 6.7 - Number of users with global QoE ≥ 3 for Y = 90, 80 and 70% for each algorithm. 66
Figure 6.8 - Percentage of satisfied users as a function of connected users streaming video (U = 3.5). 68
Figure 6.9 - Number of users with global QoE ≥ 3.5 for Y = 90, 80 and 70% for each algorithm. 68
Figure 6.10 - Percentage of satisfied users as a function of connected users streaming video (U = 4). 69
Figure 6.11 - Number of users with global QoE ≥ 4 for Y = 90, 80 and 70% for each algorithm. 70
Figure 6.12 - Percentage of users with global QoE < 2 as a function of connected users streaming video. 71

List of tables

Table 4.1 - Number of resource-block pairs available for each bandwidth. 26
Table 4.2 - Efficiencies for each CQI. 27
Table 5.1 - Perceived quality by user that experienced a global QoE equal to $QoEi$. 54
Table 6.1 - Capacity of MBF for different values of α for Z = 2, Y = 90%, U = 3 and W = 0%....... 64
Table 6.2 - Capacity of MBF for different values of α for W = 5%, Z = 2, Y = 90% and U = 3....... 64
Table 6.3 - Algorithms' capacities and gains with respect to Round Robin for U = 3 and Y = 90, 80 and 70%. 67
Table 6.4 - Algorithms' capacities and gains with respect to Round Robin for U = 3.5 and Y=90, 80 and 70%. 69
Table 6.5 - Algorithms' capacities and gains with respect to Round Robin for U = 4 and Y = 90, 80 and 70%. 70
Table 6.6 - Maximum capacities imposed by the value of W %. 72

List of acronyms

ABS	Adaptive Bitrate Streaming
ACR	Absolute Category Rating
APN	Access Point Name
BET	Blind Equal Throughput
BMFF	Base Media File Format
CDF	Cumulative Distribution Function
CLO	Cross-Layer Optimization
CPU	Central Processing Unit
CQI	Channel Quality Indicator
DASH	Dynamic Adaptive Streaming over HTTP
DCPF	Delay-Constrained Proportional Fairness
DM	Device Management
DRM	Digital Rights Management
DSCQS	Double Stimulus Continuous Quality Scale
DSIS	Double Stimulus Impairment Scale
EDF	Earliest Deadline First
FDD	Frequency Division Duplex
FIFO	First In First Out
FSIG	Frame Significance
GUI	Graphical User Interface
HDS	Adobe HTTP Dynamic Streaming
HLS	Apple HTTP Live Streaming
HOL	Head OF Line
HTTP	Hypertext Transfer Protocol
IBFF	ISO Base File Format
IEC	International Electrotechnical Commission
IP	Internet Protocol
ISO	International Organization for Standardization
LR	Loss Rate
LTE	Long Term Evolution
LWDF	Largest Weighted Delay First

MBF	Maximum Buffer Filling
MDC	Multiple Description Coding
MLBS	Mean Lost Burst Size
MOS	Mean Opinion Score
MPD	Media Presentation Description
MSE	Mean Square Error
MS-SSIM	Multi-Scale Structural Similarity
MT	Maximum Throughput
M-LWDF	Modified Largest Weighted Delay First
OFDMA	Orthogonal Frequency-Division Multiple Access
OIPF	Open IPTV Forum
PBCH	Physical Broadcast Channel
PC	Pair Comparison
PDP	Packet Data Protocol
PF	Proportional Fair
PFBF	Proportional Fair with Barrier for Frames
PSNR	Peak-Signal-to-Noise-Ratio
PSS	Primary Synchronization Signal
PUCCH	Physical Uplink Control Channel
PUSCH	Physical Uplink Shared Channel
QAAD	QoE-enhanced Adaptation Algorithm over DASH
QMC	Quality of Experience Measurement Collection
QoE	Quality of Experience
QoS	Quality of Service
RB	Resource Block
RNN	Random Neural Network
RR	Round Robin
RRM	Radio Resource Management
RTCP	Real-Time Transport Control Protocol
RTP	Real-Time Transport Protocol
RTSP	Real-Time Transport Stream Protocol
RTT	Round Trip Time
SAP	Stream Access Point

SINR	Signal-Interference plus Noise Ratio
SSIM	Structural Similarity
SSCQE	Single Stimulus Continuous Quality Evaluation
SSS	Secondary Synchronization Signal
SVC	Scalable Video Content
TCP	Transport Control Protocol
TDD	Time Division Duplex
TS	Transport Stream
TTA	Throughput To Average
TTI	Transmission Time Interval
UDP	User Datagram Protocol
URI	Uniform Resource Identifiers
VoD	Video on Demand
VoIP	Voice over IP
VSSIM	Video Structural Similarity
VQM	Video Quality Metric
XML	Extensible Markup Language

Chapter 1

Introduction

This chapter provides the scope and objectives of this thesis, and presents the motivation and the global context inspiring the problem to be solved. Finally, the thesis' main contributions and structure are also outlined.

1.1 Context and motivation

In the last years, the multimedia content consumed by mobile users has exponentially increased, particularly video content. Users are constantly downloading video streams, using video conferencing applications, or even broadcasting their own video streams to the Internet through social applications, congesting the network. Therefore, the need of quality assessment became essential since network operators want to control their network resources while maintaining a high degree of user satisfaction. The fact is that the measurement of technical parameters fails to give an account of the user experience, and therefore operators and content providers are searching for mechanisms capable of assessing such user satisfaction through Quality of Experience (QoE) techniques.

Unlike Quality of Service (QoS) metrics, such as packet loss rate, packet delay rate, packet jitter rate and throughput, that are typically used to indicate the impact on the video quality level from the network's point of view, QoE goes beyond pure technical measures. It considers the end-customer perception, who interacts with the device, its user-interface, the network, and the service behind it [1]. QoE has been defined as "the degree of delight or annoyance of a person experiencing an application, service, or system. It results from the fulfillment of his or her expectations with respect to the utility and/or enjoyment of the application, service or system in the light of the person's personality and current state" [2].

In [3], a protocol stack to form a conceptual relationship between QoS and QoE is proposed. It is shown in Figure 1.1 and has three levels: Network QoS, Application QoS and User QoS (QoE).

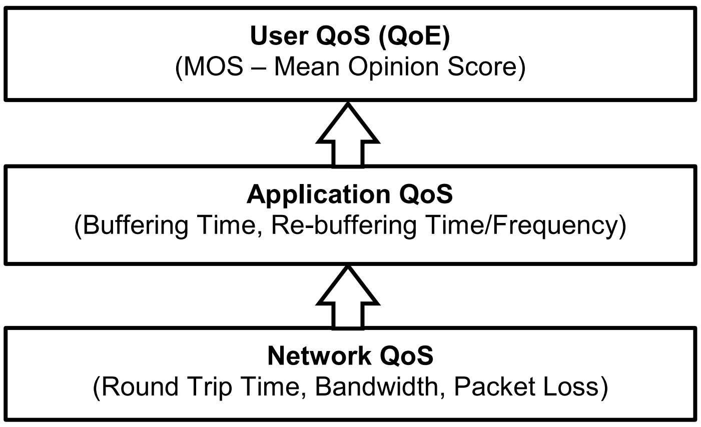

User QoS (QoE)
(MOS – Mean Opinion Score)
Application QoS
(Buffering Time, Re-buffering Time/Frequency)
Network QoS
(Round Trip Time, Bandwidth, Packet Loss)

Figure 1.1 - Three levels of QoS proposed in [3].

Network QoS is the network path performance between a server and a client. Application QoS, on the other hand, reflects the performance from an application point of view. The QoE level reflects the overall performance from the user's point of view. It is actually on the top of the stack because it depends on the performance of the levels below. QoE is usually expressed using a Mean Opinion Score (MOS) where people evaluate a video after compression and transmission with a score ranging from 1 ("Bad") to 5 ("Excellent").

Clearly users' expectations about the services provided are getting higher. Nowadays users want video streaming services that are available (they want to reach the videos they want), consistent (they don't want any re-buffering events), and high-quality (they want content that looks great). In order to satisfy all these requirements, it is needed a high-functioning data center, high-throughput and resilient networks, and powerful mobile devices that handle the video streaming along with anything else the user might be doing at the same time. Furthermore, in a wireless network there are some users experiencing bad radio channel conditions. This results in undesired effects such as long initial buffering times, video pauses (stalls), long re-buffering periods, frequent changes in video quality, frame rate drops and/or in audio/video desynchronization, degrading user's QoE. Therefore, the development of high-performance physical-layer techniques is necessary, as well as intelligent resource management strategies to provide high throughput and efficient use of resources, while maintaining a high degree of user satisfaction. Among these strategies, scheduling algorithms have an important role and a lot of research still needs to be done. There is already some developed work that can be found in the literature but it is mostly found developed scheduling algorithms that improve only the mean video quality of the video downloaded or the mean frequency of stalls, etc. Particularly, there are very few studies that analyze the capacity of the schedulers regarding the number of users that can be streaming video with a good or excellent QoE in order to achieve a certain percentage of satisfied users and the QoE fairness level between the connected users.

1.2 Objectives

One of the most challenging issues that next-generation wireless networks will face is to provide quality of service and quality of experience guarantees to the large amount of multimedia applications that are emerging every day. In this context, much work has already been developed on the network resources scheduling process. However, most of the published work do not assess the performance of the proposed scheduling algorithms regarding the number of users who have a good experience streaming video.

In this context, the main objective of this thesis is to design, implement and assess an improved solution for scheduling network resources in a LTE network scenario where several mobile clients implementing the MPEG-DASH specification are streaming video. In particular, the goal is to design a scheduler that increases the network capacity in terms of the number of satisfied users streaming video, while maintaining a high level of QoE fairness. This scheduler makes use of the reported QoE metrics by DASH clients and their radio channel quality status. The proposed solution performance is then compared to the most well-known schedulers in the literature.

1.3 Main contributions

The main contributions of this thesis are related to the way the proposed scheduling algorithm allocates network resources, based on the QoE metrics reported by the DASH clients and their channel quality status. The scheduling algorithm not only achieves high network capacities regarding the number of users who are satisfied with their streaming session, but also maintains a high QoE fairness level between all the connected users. Furthermore, the provided QoE fairness level can be adjusted according to the mode in which the scheduler operates and the mobile operator's intention.

The performance assessment of some of the most well-known schedulers in the literature in terms of the number of satisfied and non-satisfied users is also done, in a scenario where several mobile users who have time-varying radio channels that were simulated using OMNET++, are connected to a LTE network, streaming video from the server.

Another contribution is the developed simulator for the analysis of performance of scheduling algorithms, rate adaptation algorithms for requesting video segments and QoE models, in a simulated LTE network scenario. In this dissertation, the essential base information for further development and implementation is also provided.

1.4 Thesis outline

This thesis is organized in seven chapters, with this first one introducing the work in terms of context, motivation and main objectives.

Chapter 2 presents a review of the most relevant concepts of MPEG-DASH specification. Firstly, the architecture of this technology and the protocols behind it are presented; next it is described how the video content exists and is organized in the servers; finally, the available DASH profiles are presented.

Chapter 3 briefly reviews the factors that influence QoE and the QoE assessment methods. It also presents the QoE metrics that a DASH client can report to the server and the reporting procedure.

Chapter 4 presents the proposed QoE-aware scheduling algorithm for adaptive HTTP video delivery in a LTE network.

Chapter 5 describes in detail the simulator developed in the scope of this work to assess the performance of the scheduling algorithms in study.

Chapter 6 presents the performance assessment methodology and the simulation parameters. Also, the assessment results are presented and discussed.

Chapter 7 concludes this thesis with a summary and suggestions for future work.

The Appendix A presents the pseudocode of the implemented scheduling algorithms, used to make the comparison with the proposed scheduler.

Chapter 2

DASH overview

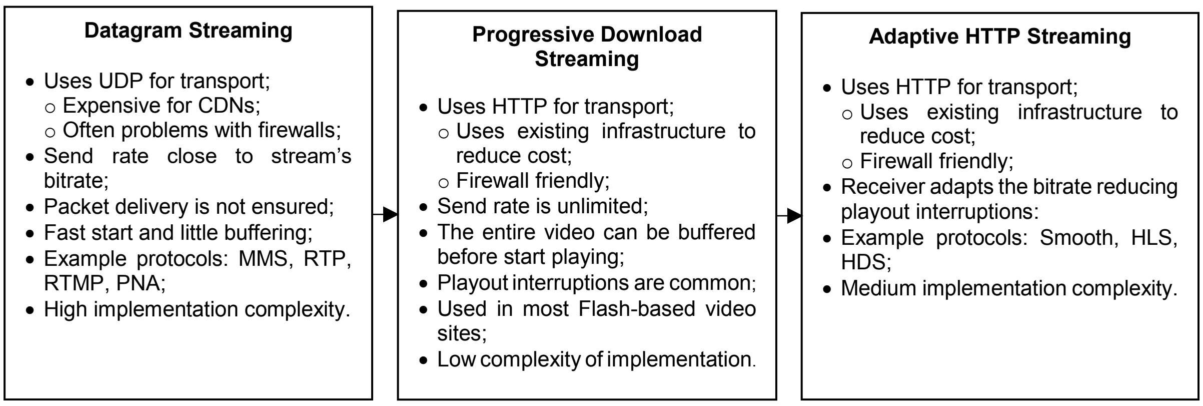
2.1 Introduction

The proliferation of powerful mobile devices and resource-demanding multimedia applications is outpacing the capacity enhancements in next generation wireless networks. Internet traffic is growing quickly mostly because users are constantly downloading video streams, using video conferencing applications, or even broadcasting their own video streams (on live or on-demand) to the Internet, congesting the network. Furthermore, when it comes to wireless networks, they face problems such as background noise, narrow frequency spectrum and varying degrees of network coverage and signal strength. These lead to loss of data or re-transmissions causing delays, degraded video quality and re-buffering events when streaming the video, affecting user's QoE. In response to that, technologies like Adaptive Bitrate Streaming (ABS) have been developed. It starts by detecting a user's bandwidth and CPU capacity in real time and adjusting the quality of a video stream accordingly. It requires the use of an encoder which can encode a single source video at multiple bitrates. The player client [4] switches between streaming the different encoded contents that exist on the servers depending on available resources. So, when the user notice that video quality is going down, this is due to bitrate adaptation – it is getting lowered – to make sure that video streaming is not interrupted. ABS is built into technologies and products like Adobe HTTP Dynamic Streaming (HDS), Apple HTTP Live Streaming (HLS) and Microsoft Smooth Streaming. However, these solutions are closed systems with its own manifest formats, content formats and streaming protocols. To overcome this lack of interoperability among devices and servers of different vendors, MPEG developed a specification with the help of many experts and in collaboration with other standards groups, such as the Third Generation Partnership Project (3GPP) and the Open IPTV Forum (OIPF). The resulting MPEG standardization of Dynamic Adaptive Streaming over HTTP is now simply known as MPEG-DASH [5]. As described in ISO/IEC 23009-1 document, MPEG-DASH can be viewed as an amalgamation of the industry's three prominent adaptive streaming protocols – Adobe HDS, Apple HLS and Microsoft Smooth Streaming.

This chapter presents the main concepts and design principles about the adaptive streaming technique called MPEG-DASH. Firstly, a brief evolution towards adaptive bitrate streaming and a background on the motivations for MPEG-DASH are given. Then the architecture is presented, as well as it is explained how media content is organized in the servers. Finally, the profiles defined by MPEG-DASH are described.

2.2 Adaptive HTTP streaming

Especially in a wireless network environment, there are problems such as background noise, narrow frequency spectrum and varying degrees of network coverage and signal strength. To overcome these problems, it would be good to adapt the video bitrate and/or other parameters to better fit the varying channel conditions. Thus, adaptive video streaming over a mobile network has been of concern in the research community, where a great number of solutions have been proposed. The evolution of video streaming towards adaptive streaming over Hypertext Transfer Protocol (HTTP) is briefly explained in Figure 2.1.


Datagram Streaming

- Uses UDP for transport;
  - Expensive for CDNs;
  - Often problems with firewalls;
- Send rate close to stream's bitrate;
- Packet delivery is not ensured;
- Fast start and little buffering;
- Example protocols: MMS, RTP, RTMP, PNA;
- High implementation complexity.

Progressive Download Streaming

- Uses HTTP for transport;
  - Uses existing infrastructure to reduce cost;
  - Firewall friendly;
- Send rate is unlimited;
- The entire video can be buffered before start playing;
- Playout interruptions are common;
- Used in most Flash-based video sites;
- Low complexity of implementation.

Adaptive HTTP Streaming

- Uses HTTP for transport;
  - Uses existing infrastructure to reduce cost;
  - Firewall friendly;
- Receiver adapts the bitrate reducing playout interruptions:
- Example protocols: Smooth, HLS, HDS;
- Medium implementation complexity.

Figure 2.1 - Evolution of streaming toward adaptive HTTP streaming.

The traditional view of video streaming was based on a server sending encoded video data to the client at the rate of consumption, thus dictated by the typically variable bitrate that derives from compression of the video data. The protocol used for communicating client requests to the server was Real-Time Transport Stream Protocol (RTSP), with Real-Time Transport Protocol (RTP) and Real-Time Transport Control Protocol (RTCP) used for carrying the data and out of band stream management information, respectively. The initial idea was to use the above protocols over User Datagram Protocol (UDP) for some reasons:

- Real time transmission;
- Avoidance of any congestion control mechanisms;
- Avoidance of transmission rate control mechanisms present in TCP connections

Web Servers use HTTP as the main communication protocol at the application layer. The existing web infrastructure, and the fact that the port number 80 used by HTTP is usually open to firewalls, are strong reasons for adopting HTTP as the communication protocol for video data, rather than those described above in traditional streaming.

Progressive HTTP download was then adopted by most video servers, such as YouTube, whereby the HTTP request by the client is serviced by the server and the client media player presents the data contained in the file while the file is being downloaded. With progressive downloading, the client does not need to download the whole file before viewing parts of it. In the case that a user pauses the media player, progressive download will continue downloading the file until it is completed, and upon the viewer resuming play, the presentation will continue. If the user chooses not to resume download, the HTTP mechanism would clearly waste bandwidth. With traditional RTSP based streaming, pausing would send a message to the server to stop sending the data [6].

Both the RTSP and HTTP progressive downloading streaming mechanisms are based on the transmission of a given video file from the server to the client. However, this is a static selection that is made before transmission starts, and once started the same file is used till the end. In fact, there is a great variability from user to user in terms of available bandwidth and mobile devices' capabilities which dictates the need for different encoding qualities and thus bitrates. Dynamic adaptation to network conditions is, therefore, an interesting way to overcome these variabilities. Adaptive bitrate streaming mechanism consists of changing the bitrate according to currently available resources (bandwidth, Central Processing Unit (CPU) load, battery charge left, screen size, etc.). The client chooses the bitrate as most of the information needed to select new bitrates is only known by the client, not the server. Often this adaptation is performed with coded-based approaches such as Multiple Description Coding (MDC) or Scalable Video Content (SVC) but they have almost no support in the market. Thus, another bitrate adaptation mechanism based on traditional codecs such as H.264 or HTTP has granted far greater popularity. This technology, known as adaptive HTTP streaming, is offered by companies like Microsoft, Apple or Adobe. Although this technology brings benefits for the content distributor (e.g., cheaper than specialized servers at each node, better scalability and reach to end network), it is the end user who gets the most out of it. Adaptive streaming supports fast start-up and seek times. Beginning at the lowest bitrate, it changes to a higher one and, as long as the user meets the minimum bitrate requirements, there should be no stalls followed by re-buffering events, disconnects or playback stutter. Adaptive HTTP streaming has become the most common transport protocol for adaptive streaming because it inherits all the benefits of progressive streaming while offering a solution to the fluctuating bandwidth problem. In Figure 2.2, a scenario is shown where the client switches smoothly to a stream with lower bitrate that exists on the HTTP Server whenever the buffer is almost empty or to a higher bitrate if the bandwidth permits it.

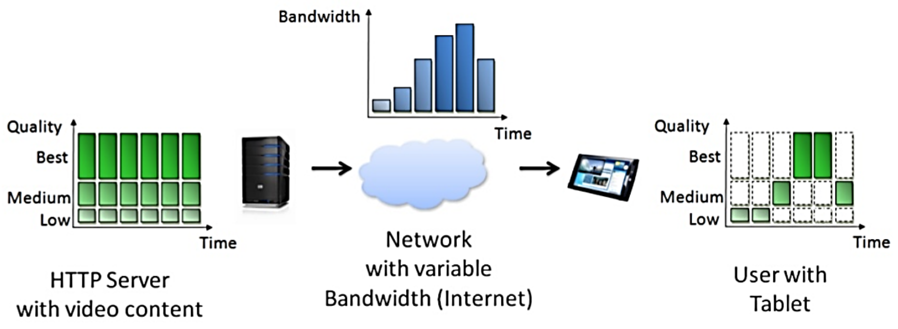


Figure 2.2 - Workflow of adaptive HTTP Streaming.

Several adaptive HTTP streaming formats are presently accessible, most significantly Apple's HTTP Live Streaming, Microsoft's Smooth Streaming and Adobe's HTTP Dynamic Streaming. These solutions are closed systems with its own manifest formats, content formats and streaming protocols. To allow interoperability, in 2009 MPEG developed a streaming standard in collaboration with many companies and organizations such as 3GPP and OIPF, giving birth to MPEG-DASH.

2.3 Scope of MPEG-DASH

Figure 2.3 illustrates a simple streaming scenario between an HTTP server and a DASH client. Only the orange blocks are defined by the specifications and are responsible for creating content and handle metadata, such as resources location and format. The green elements are outside of DASH scope. So, the delivery of the metadata file and media encoding formats containing the segments is not specified. This also applies to the client's action for fetching, adapting and playing the content [7].

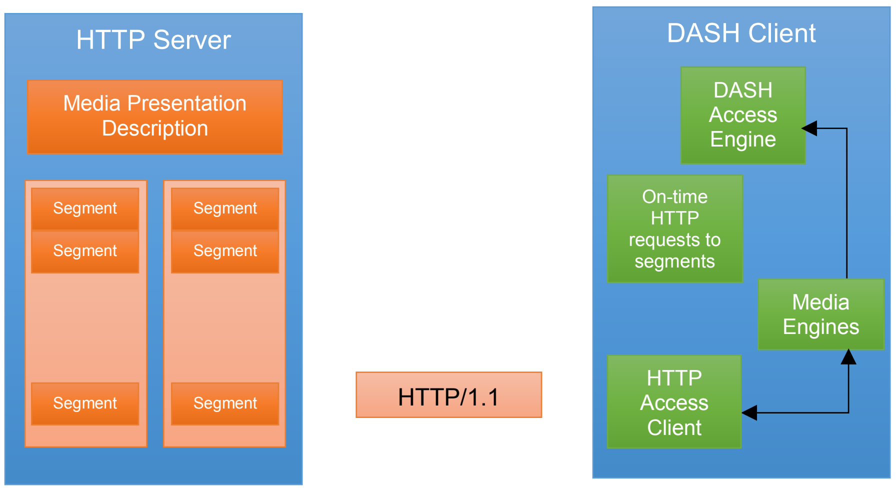


Figure 2.3 - MPEG-DASH system description.

The server holds variable encoded media data of the same content that is formed by a content creator generator. DASH specifies how content is produced and how metadata is handled. The content exists on the server in two parts that are stored into a regular web server:

- **Segments**: contain the actual multimedia bit streams in the form of chunks, in single or multiple files.
- **Media presentation description (MPD)**: a XML document that describes the temporal and structural relationships between segments. It provides sufficient information for the DASH Client to provide a streaming service to the user by requesting Segments from an HTTP server and demultiplexing, decoding and rendering the included media streams. It describes the content that is available in the server, including the URL addresses of stream chunks, byte-ranges, different bitrates, resolutions, and content encryption mechanisms.

DASH does not prescribe any client-specific playback functionality; rather, it just addresses the formatting of the content and associated MPD [8]. To play the content, the DASH client first obtains the MPD usually using the HTTP. The MPD allows the user to learn about the program timing, media-content availability, media types, resolutions, minimum and maximum bandwidths, and the existence of various encoded alternatives of multimedia components, accessibility features and required digital rights management (DRM), media-component locations on the network, and other content characteristics. Using this information, the DASH client selects the appropriate encoded alternative and starts streaming the content by fetching the segments using HTTP GET requests. After appropriate buffering to allow for network throughput variations, the client continues fetching the subsequent segments and also monitors the network bandwidth fluctuations. Using these measurements, the DASH client decides how to adapt to the available bandwidth by fetching segments of different alternatives (with lower or higher bitrates) to maintain an adequate buffer occupation [7].

2.4 DASH client model

Figure 2.4 illustrates the logical components of a conceptual DASH client model.

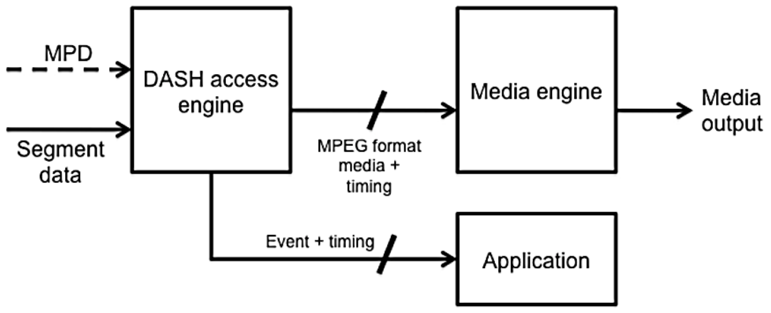


Figure 2.4 - DASH client model [9].

The DASH access engine receives the MPD, constructs and issues requests and receives Segments or parts of Segments. The output of the DASH access engine consists of media in MPEG container formats (ISO/IEC 14496-12 ISO Base Media File Format or ISO/IEC 13818-1 MPEG-2 Transport Stream), or parts thereof, together with timing information that maps the internal timing of the media to the timeline of the Media Presentation. In addition, the DASH access client may also receive and extract Events that are related to the media time. The events may be processed in the DASH client or may be forwarded to an application that is being executed. Examples for events are indication of MPD updates on the server, possibly providing the detailed update as part of the messages. The event mechanisms may also be used to deliver media time related application events, for example information about ad insertion, etc.

To play the multimedia file, the client player uses an URL to request the MPD file. By analyzing the MPD, the DASH client determines what are the viable options for this particular stream in terms of stream duration, bitrates available, location of the media, required DRM, resolutions. Using this information, the client selects the appropriate encoded alternative and starts streaming the content by fetching the segments using HTTP GET requests, as it can be seen in Figure 2.5.

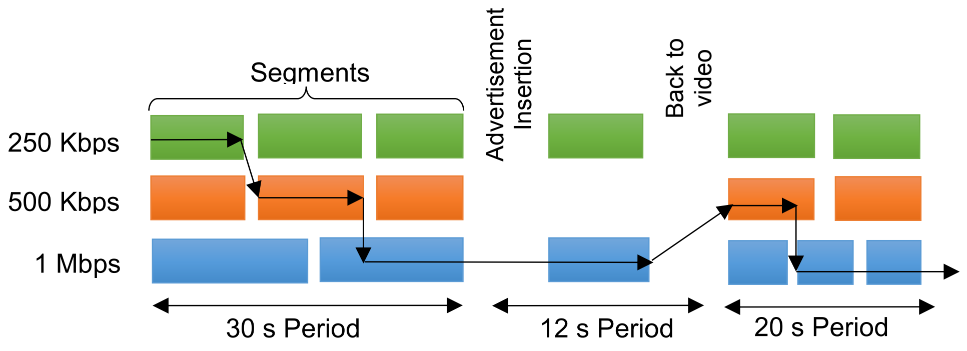


Figure 2.5 - Client streaming process.

The client entirely controls the delivery of services. After learning the information provided by the MPD the player client chooses which adaptive stream bitrate and resolution to play and switches to different bitrate streams depending on available resources: buffer conditions, network conditions, user change in resolution (e.g. full screen), device activity, etc.

2.5 DASH data model

A Media Presentation is a collection of data that is accessible to a DASH Client to provide a streaming service to the user. It is described by a MPD file that contains metadata required by the client to construct appropriate HTTP-URLs to access Segments and to provide the streaming service to the user. DASH is based on the hierarchical data model represented in Figure 2.6.

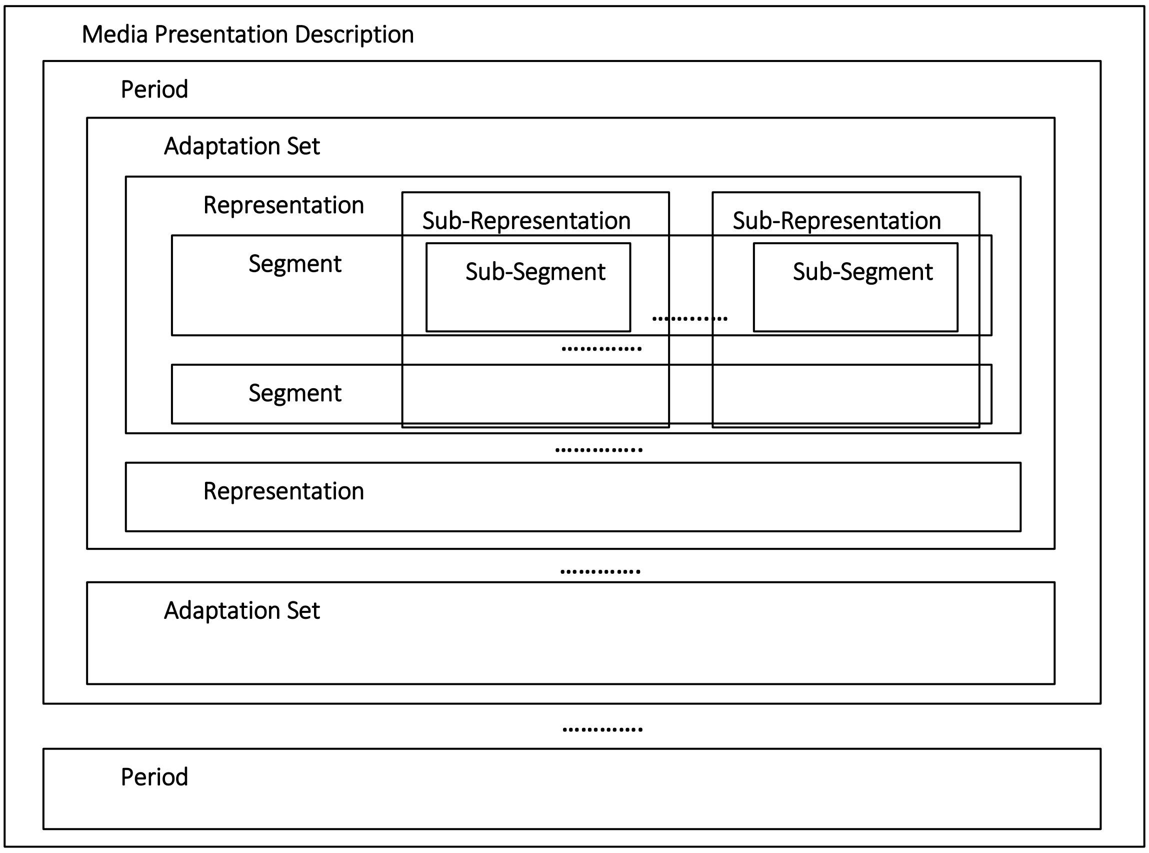


Figure 2.6 - DASH data model [8].

This means:

- A Media Presentation consists of a sequence of one or more consecutive non-overlapping Periods.
- Each Period contains one or more Adaptation Sets which include one or more Representations from the same media content.
- Each Representation consists of one or more Segments.
- Segments contain media data and/or metadata to decode and present the included media content.

A. Period element

The MPD describes the sequence of periods in time that make up the Media Presentation. Each MPD element consists of one or more Periods. A Period has a start time and typically represents a media content period during which a consistent set of encoded versions of the media content is available i.e. the set of available bitrates, languages, captions, subtitles etc. does not change during a Period. Nevertheless, the client can adapt during a Period according to bitrates, resolutions and available codecs for that given Period. In addition, a Period might be used to separate content – insert an ad, change the angle of a live transmission – for example, if there is a need to introduce the ad exclusively in high definition while the rest of the content is available in a larger range of resolutions, it is just a matter of establishing a single Period for the ad containing only a high resolution [8].

B. Adaptation Set element

Within a Period, material is arranged into Adaptation Sets that allow to form logical groups with different components. An Adaptation Set represents a set of interchangeable encoded versions of one or several media content components. For example, there may be one Adaptation Set for the main video component and a separate one for the main audio component. If there is other material available, for example captions or audio in other languages, then these may each have a separate Adaptation Set [8].

C. Representation element

An Adaptation Set contains alternate Representations, i.e. only one Representation within an Adaptation Set is expected to be presented at a time. Different Representations in One Adaptation Set represent perceptually equivalent content but are encoded in different ways – codec, resolution, bandwidth, etc. Typically, this means that clients may switch dynamically from Representation to Representation within an Adaptation Set in order to adapt to network conditions or other factors. The Adaptation Set and the contained Representations shall be prepared and contain sufficient information such that the switching across different Representations in one Adaptation Set is seamless. Switching refers to the presentation of decoded data up to a certain time t, and presentation of decoded data of another Representation from time t onwards.

Representations may also include Sub-Representations to describe and extract partial information from a Representation. The Sub-Representation element describes properties of one or several media content components that are embedded in the Representation. It may for example describe the exact properties of an embedded audio component (e.g., codec, sampling rate, etc.), an embedded sub-title (e.g., codec) or it may describe some embedded lower quality video layer (e.g. some lower frame rate, etc.) [8].

D. Segment element

Within a Representation, the content may be divided in time into Segments for proper accessibility and delivery. In order to access a Segment, each one is described by an URL. Consequently, a Segment is the largest unit of data that can be retrieved with a single HTTP request. Segments are assigned a duration, which is the duration of the media contained in the Segment when presented at normal speed. Typically, all Segments in a Representation have the same duration. However, Segment duration may differ from Representation to Representation. Segments may be further subdivided into Sub-Segments each of which contains a whole number of complete access units. If a Segment is divided into Sub-Segments they are described by a compact Segment index within a Media Segment separately from MPD, which provides the presentation time range in the Representation and corresponding byte range in the Segment occupied by each Sub-Segment. Clients may download this index in advance and then issue requests for individual Sub-Segments [8].

DASH does not impose a specific length for segments. Larger segments allow using less overhead bits as each segment needs to be requested by HTTP and each request demands an overhead. On the other hand, smaller segments are used for live situations or high-varying bandwidth scenarios as they allow faster transitions between bitrates [7] [8].

Figure 2.7 represents a possible composition of a Media Presentation, which includes several Periods, Adaptation sets and Media Segments. The Periods slice the video over time. The adaptation sets separate the content logically. The Representations represent the same period of content but with different characteristics (e.g. bandwidth) and finally the segments divide the content in shorter periods of time to enable the client for proper accessibility and delivery. Usually, segmentation produces an initialization segment that contains metadata which is necessary to present the media streams encapsulated in Media Segments.

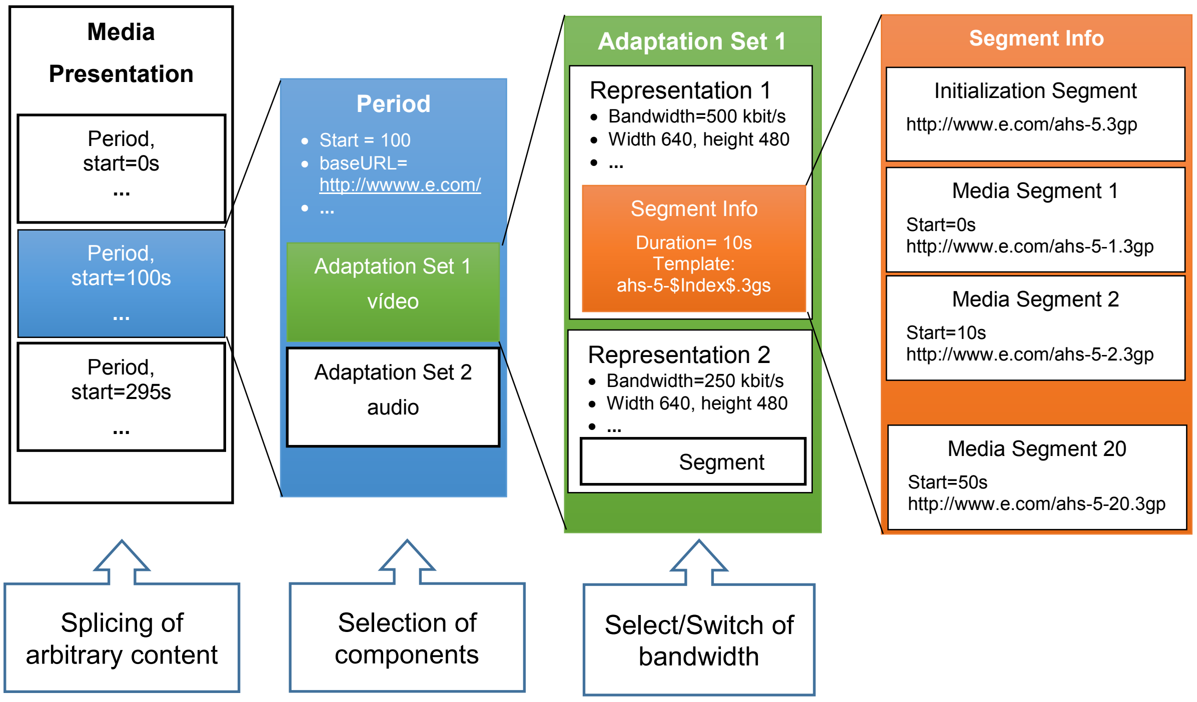


Figure 2.7 - DASH data model representation.

2.6 Protocols

Figure 2.8 shows a protocol stack for services in the context of the 3GPP DASH specification [9].

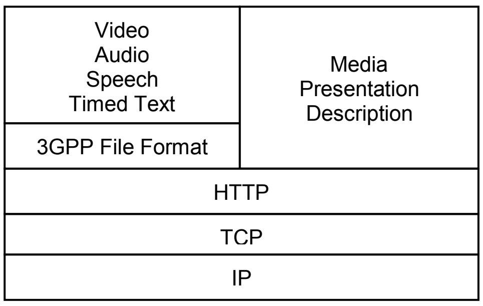


Figure 2.8 - Overview of the protocol stack [9].

The use of HTTP as an application layer protocol provides some advanced features such as caching, redirection or authentication. DASH clients should use the HTTP GET method or the HTTP partial GET method, as specified in RFC 2616 [10], clause 9.3, to access media offered at HTTP-URLs. The use of Transport Control Protocol (TCP) provides reliable, ordered, and error-checked delivery of a stream of octets between the server and the user. At the lower levels of the protocol stack, due to network congestion, traffic load balancing, or other unpredictable network behavior, IP packets may be lost, duplicated, or delivered out of order. TCP detects these problems, requests re-transmission of lost data, rearranges out-of-order data and even helps minimize network congestion to reduce the occurrence of the other problems. Once the TCP receiver has reassembled the sequence of octets originally transmitted, it passes them to the receiving application [11]. This way the media content has mathematically the same quality in the server and in the client side. However, the accuracy and reliability provided by TCP is paid with a higher delay when delivering content.

2.7 Media content

2.7.1 Format

In order to deliver media content, firstly it needs to be compressed to reduce its size during transportation, over Internet. To do so, it is used a codec, usually MPEG-1, MPEG-2, WMV or H.264 for video and AAC for audio. Once compressed, the content is packaged in containers to be transported to the end user. The most familiar containers are MP4, MPEG-2 TS and Flash. Under the MPEG DASH standard, the media segments can contain any type of media data. However, the standard provides specific guidance and formats for use with two types of segment container formats – MPEG-2 Transport Stream (MPEG-2 TS) and ISO base media file format (ISO BMFF). MPEG-2 TS is the segment format that HLS currently uses, while ISO BMFF (which is basically the MPEG-4 format) is what Smooth Streaming and HDS are using [5]. Finally, the codec will be also responsible for decompressing the content on the end user side.

2.7.2 Segmentation

The media content must be firstly segmented according to what was seen in section 2.5 and stored on servers so it can be requested by the users. The segmentation can result into multiple separated segment files or just one segment file with multiple sub-segments. Usually, segmentation produces an initialization segment (as it is represented in Figure 2.6) which initializes the media for playback. It contains metadata that is necessary to present the media streams encapsulated in Media Segments. This segment contains the stream access points (SAP), marked frames within the streaming, allowing the segment switching on the playback. A SAP is a position in a Representation enabling playback of a media stream to be started using only the information contained in Representation data starting from that position onwards. If there is not an initialization segment for a Representation, it means that each media segment within the Representation shall be self-initialized.

Segmentation and sub-segmentation may be performed in ways that make switching simpler. For example, in the very simplest cases each Segment or Sub-Segment begins with SAP and the boundaries of Segments or Sub-Segments are aligned across the Representations of one Adaptation Set. In this case, switching Representation involves playing to the end of a (Sub) Segment of one Representation and then playing from the beginning of the next (Sub) Segment of the new Representation. Also the Media Presentation Description and Segment Index may make switching simpler as they provide various indications, which describe properties of the Representations [8].

2.7.3 Reproduction

The content exists on the server in two parts: the MPD and the segments that contain the multimedia itself. Firstly, the client requests the MPD and then he parses it in order to learn about the program timing, media-content availability, media types, resolutions, minimum and maximum bandwidths, and the existence of various encoded alternatives of multimedia components, etc., so he can select a collection of Adaptation Sets suitable for its environment. Within each Adaptation Set it selects one Representation, typically based on the value of an attribute of that Representation, called *Bandwidth*, but also taking into account client decoding and rendering capabilities. The value of the attribute bandwidth is the bitrate of the channel (supposing it is constant) that a client needs, to have enough data for continuous playout.

To access the content, firstly the initialization segment is requested (if there is one) and then the media ones. The client buffers a certain amount of media before starting the presentation. The content playback will begin as soon as it reaches a minimum amount of data. These requested files should be of low quality to minimize the initial buffering time and start playing the content as fast as possible. Once the presentation has started, the client continues consuming the media content by continuously requesting Media Segments or parts of Media Segments. The client may switch Representations taking into account updated MPD information and/or updated information from its environment, e.g. change of observed throughput. If a certain frame that was not yet downloaded is reached, the playback will stop (stall) and a re-buffering event will begin, degrading the QoE of the user.

2.8 DASH profiles

DASH specify profiles to enable interoperability and the signaling of the use of features. In Figure 2.9, the six available profiles defined by DASH are represented, each one addressing specific applications/domains. Each profile imposes a set of specific restrictions. Those restrictions are typically on features of the MPD document related to encoding of the segments and the source of the stream (it can be live or On Demand) and on Segment formats. The restrictions may be also on content delivered within Segments, such as on media content types, media formats, codecs, and protection formats, or on quantitative measures such as bitrate, Segment duration and size, as well as horizontal and vertical visual presentation size. The MPD has an attribute, called *Profiles*, which specifies a list of Media presentation profiles.

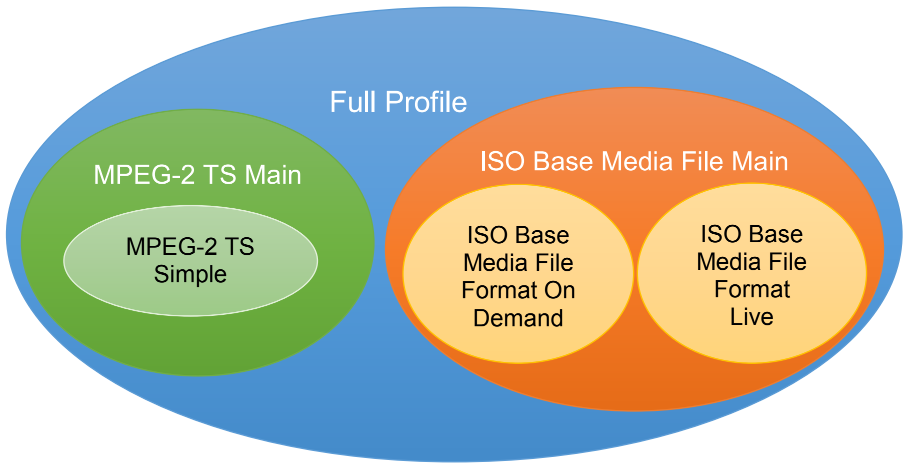

Full Profile
MPEG-2 TS Main
MPEG-2 TS Simple
ISO Base Media File Main
ISO Base Media File Format On Demand
ISO Base Media File Format Live

Figure 2.9 - MPEG-DASH profiles as defined in ISO/IEC 23009 [8].

There are three profiles defined relying on the **ISO Base File Format (IBFF)** encoded files [8]:

- **ISO Base media file format On-Demand** – Intended to provide basic support for on-demand content, supporting large Video on Demand (VoD) libraries with minimum amount of content management. The primary constraints imposed by this profile are the requirement that each Representation is provided as a single Segment, that Sub-segments are aligned across Representations within Adaptation Set and that Sub-segments must begin with SAPs. This allows scalable and more efficient use of HTTP servers and simplifies seamless switching.
- **ISO Base media file format Live** – Optimized for live encoding and low latency by encoding and immediate delivery of short Segments consisting of one or more movie fragments of ISO file format. Each fragment may be requested as soon as available, so usually it is necessary to request MPD update prior to each Segment request. Despite this profile is optimized for live services, it may work as well as an On-Demand one in case the MPD@Type attribute is set to 'static'.
- **ISO Base media file format Main** – Support for both On Demand and Live content. ISO Base media file format Live ISO Base media file format On-Demand are a subset of this profile.

There are also two profiles related to **MPEG-2TS** encoded files [8]:

- **MPEG-2TS Main** – Imposes little constraints on the Media Segment format for MPEG-2 Transport Stream content.
- **MPEG-2TS Simple** – It is a subset of MPEG-2TS Main profile. It poses more restrictions on the encoding and multiplexing in order to allow simple implementation of seamless switching.

The previous profiles simplify the content organization of the metadata file according to the encoding of the segments. There is also a sixth profile called the **Full Profile** which supports both ISO Base and MPEG-2 TS media file formats. This profile includes all features and Segment Types defined in ISO/IEC 23009.

A DASH client in agreement to a specific profile is only required to support those features and not the entire specification. It is also important to mention that external organizations may define themselves profiles by posing restrictions, permissions and extensions. However, it is recommended that such external profiles be not referred to as profiles but as an Interoperability Point.