TECHNISCHE UNIVERSITÄT MÜNCHEN
Lehrstuhl für Medientechnik

Adaptive Streaming of Scalable Video in Mobile Networks

Ktawut Tappayuthpijarn, M.Sc.

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Die Dissertation wurde am 13.02.2012 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 07.01.2013 angenommen.

Abstract

This thesis proposes and compares different streaming architectures for adaptive video transmission in mobile networks. The described solutions address the challenges found for both timestamp-based and progressive download video streaming paradigms and have been designed to work with the Scalable Video Coding extension of the H.264 video coding standard. For the introduced timestamp-based solution, the amount of available radio resources for streaming users is regulated according to the observed congestion situation in the mobile cell. Only minimal cross-layer communication between users' devices and the adaptation server is required to perform coordinated Rate-Distortion optimized adaptation of multiple streaming users. No cross-layer operation is required when the adaptation is to be done in an uncoordinated manner. For the solution that is based on the progressive download, a statistical model of the best-effort TCP throughput over a mobile channel has been developed. It is used to make appropriate adaptation decisions entirely by the mobile user without any cross-layer operation. The design of the latter architecture ensures its compatibility with the on-going Dynamic Adaptive Streaming over HTTP standard.

Zusammenfassung

Diese Arbeit unterbreitet Architekturvorschläge für adaptives Video-Streaming in Mobilfunknetzen und betrachtet dabei sowohl echtzeit- als auch abrufbasierte Video-Streaming Szenarien. Die vorgeschlagenen Ansätze basieren auf der skalierbaren Erweiterung des H.264-Standards. Für echtzeitbasiertes Multi-User Streaming werden die Radio-Ressourcen als Funktion der Zellauslastung zugeteilt. Dabei wird nur minimale schichtenübergreifende Information zwischen den mobilen Endgeräten und dem Adaptierungs-Server ausgetauscht. In einem modifizierten Ansatz wird die Anpassung unkoordiniert durchgeführt und somit der Einsatz von schichtenübergreifendem Informationsaustausch gänzlich vermieden. Für die abrufbasierte Architektur wird ein statistisches Modell für die Abschätzung des Durchsatzes im Falle der Verwendung von TCP/IP ohne Servicegarantien entwickelt. Dieses Modell wird verwendet, um angemessene Adaptierungsentscheidungen im mobilen Endgerät durchzuführen. Der vorgestellte Architekturvorschlag ermöglicht Kompatibilität mit dem zurzeit in MPEG entwickelten Standard für adaptives Streaming über HTTP.

Contents

Chapter 1

Introduction

1.1 Motivations

In recent years, there have been rapid deployments of high-speed 3G/4G mobile networks in many countries around the world as well as expansions of existing ones. As of 2010, there were more than 500 3G networks in operation while 300 more 4G networks were being planned worldwide, allowing billions of people easy access to broadband mobile internet [1]. Together with the widespread and increasing popularity of smart phones and tablet PC's, the volume of mobile internet traffic where a significant portion of which belongs to mobile video streaming has gone up dramatically in the last decade. In fact, a recent study by Cisco Systems, Inc. predicts that from 2010 to 2015, the mobile video streaming traffic will have on average a 100% rate of growth by volume annually and will contribute more than 60% to the overall mobile internet traffic by 2015 [2]. This significant growth rate is driven by the popularity of video sharing and hosting websites such as Youtube, embedded video contents in the news and articles, subscribed online TV, video conferencing and other foreseeable similar services in the future.

However, providing video streaming services to mobile users poses additional challenges compared to those in fixed and stationary last-mile networks. A video streaming user in a 3G/4G mobile environment is subjected to rapid changes in reception quality and congestion level in a mobile cell which affect the network's performance and the mobile user differently depending on the type of service provided. For a user that receives certain Quality of Service (QoS) guarantees from the network, e.g., a minimum guaranteed bitrate, more radio resources must be used to provide additional protection against the channel in order to maintain a stable throughput when the user is in a bad reception area. This results in inefficient resource utilization and a lower overall cell throughput. The minimum guaranteed bitrate itself can still be violated if the cell is heavily congested as this is usually only a soft guarantee. On the contrary, a mobile user that is served in a best-effort manner by the network suffers a sporadic and highly fluctuating throughput, depending largely on its reception quality relative to other users. The latter is a consequence of a more efficient resource allocation policy from the network's point of view which gives a significant weight on the users' channel quality and less on the different priorities and their QoS requirements. Regardless of whether the user has the QoS-guaranteed service or the best-effort service, packet losses, degraded video quality and/or playback interruptions are more likely in this environment, thus lowering the user's satisfaction for the streaming application. Additionally, the network operator also has to balance between providing good service quality and maximizing the network's capacity as well.

The ability to adapt the video bitrate and/or operating parameters of the underlying communication layers to better suit the varying channel and congestion situations in the cell is a promising solution to provide good quality video streaming services in a mobile environment. Adaptive video streaming over a mobile network therefore has long been of interest in the research community where a rich body of works and various solutions have been proposed. These works can be classified mainly into two broad categories, namely timestamp-based adaptive video streaming and progressive download adaptive video streaming.

For the timestamp-based architecture, the transmission rate of the video closely follows the encoding rate of the media itself. Many of the initial works perform the bitrate adaptation by using a computationally-intensive transcoding operation, temporal scalability or switching between multiple representations of the content at various qualities and bitrates. With the advent of the Scalable Video Coding (SVC) extension to the H.264 encoding standard [3], the bitrate of a SVC-encoded video can be easily adapted by adding or removing scalable layers to and from the bitstream without performing high-complexity operations. A large number of the later works therefore have exploited this feature of the SVC as well. Other solutions propose various cross-layer design concepts, such as employing Unequal Error Protection (UEP) schemes at the Physical and Link layers to offer different protection levels to different parts of the video based on their importance, or using channel indicators from the lower layers to influence video encoding parameters at the Application layer. Nevertheless, there have not been significant real-world deployments of these proposals up to date. Commercial timestamp-based streaming services, such as those developed using the Darwin Streaming Server or the Helix Universal Media Server based on the RTP/RTSP protocols [4,5], are mostly non-adaptive or only statically adapt to the terminal's capability during session setup. An example of such services is the Apple, Inc.'s QuickTime TV network [6]. Some of the reasons many timestamp-based adaptive streaming proposals have not been widely implemented, amongst other reasons, are the complications from having non-standardized cross-layer communications and interfering with the internal operations of different layers and network nodes.

For the video streaming with the progressive download architecture, the transmission rate from the server depends on the available best-effort throughput the network can provide and considered to be fair to other users. With the improvements in the last-mile mobile network technologies such that the best-effort throughput has become more practical for the demanding video streaming services, this streaming technique has gained more interest from both the industry and the research community lately. Some examples are the video players based on the Adobe Flash Player [7] or web browsers with HTML5 or later which supports progressive video streaming with HTTP [8]. Additionally, the on-going Dynamic Adaptive Streaming over HTTP (DASH) standard [9] is expected to be widely accepted. It defines means to store and retrieve video contents via HTTP while being open for research and development of smart adaptive streaming engines to drive the adaptation. Although there has been an increasing number of research works in this field recently, none of which, to the best of my knowledge at the current time of writing, has directly addressed the challenges found in a mobile environment, and proposed an adaptation engine that both utilizes the SVC and performs a Rate-Distortion optimized adaptation based on characteristics of the mobile channel. More detailed reviews of these related works as well as the current cutting-edge adaptation architectures can be found in Section 2.3.

As has been briefly elaborated, there has not been a significant deployment of a truly dynamic adaptive video streaming architecture for either the timestamp-based or the progressive download paradigms to date, especially the one for a mobile environment. Additional research works to contribute to the body of knowledge in this field and alternative practical design solutions for such adaptive architectures would be of great interest.

1.2 Contributions and Outlines

The goals of this work are to develop architectures for adaptive video streaming over a mobile network for both the timestamp-based and progressive download paradigms, and to have thorough understanding of their performances and relationships with other influential system parameters and the environments in which they operate. The proposed solutions have been designed taking into consideration the underlying principles of how the 3G/4G radio resources are shared amongst users in the cell and the fairness to other non-streaming users while in the same time being generic and independent of any specific mobile technology. The work also focuses toward exploiting the scalable property of the H.264/AVC and its SVC extension for bitrate adaptation and avoiding the expensive transcoding or having multiple representations of the video.

A short summary and contributions of the remaining chapters in this thesis is as follows.

Background and Current State of the Art

In Chapter 2, preliminaries on the related subjects are provided. This includes backgrounds on the mobile technologies of interest, how radio resources are managed and types of data service usually available to video streaming users in such networks. Overall design concepts and characteristics of the H.264/AVC video encoding standard as well as its SVC extension, video quality evaluation and performance metrics used in the later chapters are also explained. One of these metrics is an estimation of the upper bound of the Mean Opinion Score (MOS), properly weighted to take the negative effects from playback interruptions into account. The later half of this chapter discusses both the video streaming paradigms - timestamp-based and progressive download - as well as the related works and the current state of the art in detail where the advantages and disadvantages in different usage scenarios are identified.

Timestamp-Based Adaptive Video Streaming Architecture

Chapter 3 proposes a timestamp-based adaptive video streaming architecture for 3G/4G mobile technologies as well as to exploit the scalable property of the SVC. The video bitrates of the streaming users in the same mobile cell are dynamically adapted to individual channel conditions and the characteristics of their videos in a coordinated manner by means of adding and removing enhancement layers. Coordinated adaptation allows better Rate-Distortion optimized performance and increased robustness against unfavorable channel situations. Alternatively, the proposed architecture can also be simplified for uncoordinated adaptation, but with slight losses in video quality and its tolerance against the channel. It employs a congestion control mechanism to adjust the available resource budget only for participating streaming users based on simple Round Trip Time (RTT) measurements. The adaptation server does not need to know the exact amount of resources consumed by other traffic classes nor to include them into the optimization problem. The only cross-layer information required is a periodic report on resource consumption obtainable at the streaming user, or none at all if the architecture is simplified to its uncoordinated form. Other required network statistics can be easily measured at the Application layer. This reduces the dependency on extensive cross-layer information exchange between, e.g., the base station and the optimization server, and the overall complexity of the architecture compared to other similar works. In addition, the base station can remain as a transparent network node and is decoupled from the adaptation process. It is therefore possible for the optimization server to be owned and operated by other service/content providers beside the mobile network operator as well.

The coordination gain, defined as the gain in the overall video quality from doing coordinated adaptation compared to individual uncoordinated adaptation, is studied and presented in detail in this chapter. Although it is intuitive that this gain exists, this work contributes further by providing in-depth mathematical analyses and simulation results regarding its relationships with other influential system parameters, e.g., the characteristics of the videos, the number of users, the base station's resource scheduler, the used optimization algorithm, etc. These insights into various factors that affect the coordination gain are general and can be applied to other similar works as well in designing an adaptive streaming architecture in the future.

Progressive Download Adaptive Video Streaming Architecture

Chapter 4 contributes further by introducing a progressive download architecture for adaptive video streaming specifically designed for a mobile environment and is compatible with both the DASH and the SVC standards. A statistical model of the TCP throughput over a mobile channel which can be used to accurately estimate success probability of data transfer within a limited period of time has been developed and described in detail. The model does not require any cross-layer operation, but instead all necessary measurements to estimate the channel can be done entirely by the streaming application at the user. With the latest statistical information of the channel from this model, a low-complexity algorithm to determine the best strategy to request for different parts of a SVC-encoded video from a HTTP server has been constructed. Finally, simulation results from the non-adaptive streaming, adaptive streaming with multiple representations of the non-scalable AVC videos and adaptive streaming using the introduced algorithm are presented and discussed.

Conclusion and Outlook

In the last chapter, a brief summary of major contributions, the strengths and weaknesses of our proposed solutions, their performances from simulations as well as further interesting research areas are discussed.

Parts of this dissertation have been published in [10–12]

Chapter 2

Background and State of the Art

2.1 Preliminary on Mobile Technologies

The cellular mobile technologies of interest in this work are those that can provide data service, in addition to the traditional voice service, with reasonable average data rate for video streaming. Generally, these are technologies that at least meet the International Mobile Telecommunications-2000 (IMT-2000) requirements for the third generation (3G) mobile technology as defined by the International Telecommunication Union (ITU) [13]. One of these requirements is that the technology must support mobile data service with peak download speed of at least 200 Kbps - a practical bitrate range for low to medium quality video streaming. The Universal Mobile Telecommunications System (UMTS) standard as defined by the 3rd Generation Partnership Project (3GPP) is an example of such technologies which provides the maximum downlink bitrate of 384 Kbps [14]. Later improvements to this standard in Release 5, 7 and 8, commonly referred to as the High-Speed Packet Access (HSPA), the Evolved HSPA and the Long Term Evolution (LTE) respectively [15-17], can provide maximum bitrate in the range of up to tens or even hundreds of Mbps in the case of LTE with ideal channel conditions. In 2009, the ITU issued the IMT-Advance requirements for the fourth generation (4G) mobile technology [18] which calls for maximum downlink speed of at least 100 Mbps in a high mobility case and 1 Gbps in a low mobility case. Presently, only the latest 3GPP's Release 10 [19] and the amendment e of the IEEE 802.16 standard [20], referred to as the LTE Advanced and the Mobile Worldwide Interoperability for Microwave Access (Mobile WiMAX) respectively, meet these requirements and are formally qualified as 4G.

2.1.1 Radio Resource Management

Radio resources in a mobile cell are arranged differently in different technologies, depending on their underlying air interfaces. For those technologies using the Wideband Code Division

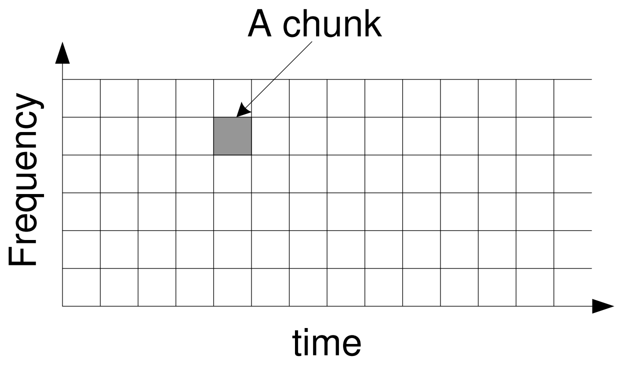


Figure 2.1: An example of radio resource chunks for a typical OFDMA technology

Multiple Access (W-CDMA) such as the Evolved HSPA and earlier technologies, the radio resources are the available transmission timeslots. The total amount of available timeslots in a given period of time is inversely proportional to the duration of each timeslot, referred to as the Transmission Time Interval (TTI) which can be from 10 ms for the UMTS to as short as 2 ms for the HSPA, depending on the processing power and desired responsiveness of the system to a varying channel situation. Alternatively, the radio resources are in the form of thousands of small temporal-frequency units in a given time period, referred to as "radio chunks" from now on, for those technologies with the Orthogonal Frequency Division Multiple Access (OFDMA) as the air interface such as the LTE and the WiMAX. In such case, a radio chunk occupies a narrow frequency band for a single sub-carrier and lasts for one TTI period which is in the range of 1-2 ms for both the LTE and WiMAX. An example of how the radio bandwidth is partitioned into radio chunks is given in Figure 2.1.

Regardless of the way resources are partitioned, these radio resources are dynamically assigned to different users in each TTI by a resource scheduler. The policy for resource assignment depends on the type of the scheduler used which could take into account, e.g., the current channel situations and QoS requirements as one of its decision metric. For the UMTS, this functionality is located at the Radio Network Controller (RNC) node. However, later standards have moved the resource scheduler closer to or at the base station to reduce the latency and improve the responsiveness to the highly-fluctuating mobile channel. Once the resource assignment is done, the Adaptive Modulation and Coding (AMC) technique is applied further to match the instantaneous channel conditions of different users with the appropriate levels of protection. This includes variations in the modulation scheme, e.g., from QPSK to 64QAM and the channel coding rate. This varies the amount of application data that can be carried and the protection bits in each radio resource unit.

2.1.2 Proportional Fair Resource Scheduler

The Proportional Fair (PF) scheduler and its variants are usually the preferred scheduler of choice to provide QoS-guaranteed services such as video streaming in a mobile network [21]. This is because the scheduler takes the average previous throughput, channel quality and other QoS parameters into consideration when assigning resources, thus providing a good balance between fairness and maximizing the throughput amongst users.

For each radio resource unit, whether it is a transmission timeslot of a W-CDMA system or a radio chunk of an OFDMA system, the PF scheduler compares the metric values from all users with respect to that particular unit and assigns it to the user with the highest metric. Assume that there are N mobile users in a cell where each user is labeled by an index $n = 1, 2, \dots, N$ and let $k = 1, 2, 3, \dots$ be an index for the current resource unit under consideration. The estimated channel quality for a user n at the current resource unit k can be, e.g., the estimated amount of data (in Bytes) that can be carried by this unit for this user. This estimation, denoted as $\tilde{\rho}_{n,k}$, is usually done at the mobile terminal and reported back to the scheduler periodically. Finally, let $\tilde{r}_{n,k}$ be the average throughput up to the time of chunk k of user n. The metric for this user on this resource unit, denoted by $\tilde{m}_{n,k}$, is defined as follows.

$$\tilde{m}_{n,k} = \frac{\tilde{\rho}_{n,k}}{\tilde{r}_{n,k}} \cdot F_{QoS} \cdot F_{delay} \tag{2.1}$$

Thus, a user with a relatively good channel and/or which has been deprived of throughput for a long time is more likely to be given this resource unit than the others. The F_{QoS} term represents a correction factor based on the QoS settings, e.g., the guaranteed bitrate (GBR) and its priority. Similarly, F_{delay} is another QoS-related correction factor that takes the queuing delay of this user into account. Although these correction terms can be different from one implementation of the PF scheduler to another, their basic properties remain similar. Specifically, the F_{QoS} tends to grow larger as the average bitrate of the user is less than the GBR and to be smaller as the average bitrate is greater than the GBR. Similarly, the F_{delay} becomes greater as the queuing delay grows larger than the guaranteed delay. The same is also true for the F_{delay} in the opposite direction.

2.1.3 Best-Effort and QoS-Guaranteed Services in Mobile Networks

Mobile network operators can generally provide data services for video streaming users in either a best-effort manner or with some degrees of QoS guarantee. For the best-effort services, the resources are allocated to users neither taking into account individual QoS requirements nor having any preference toward any user in particular. In case the PF scheduler is employed, this is equivalent to setting $F_{QoS} = F_{delay} = 1$. On the contrary,

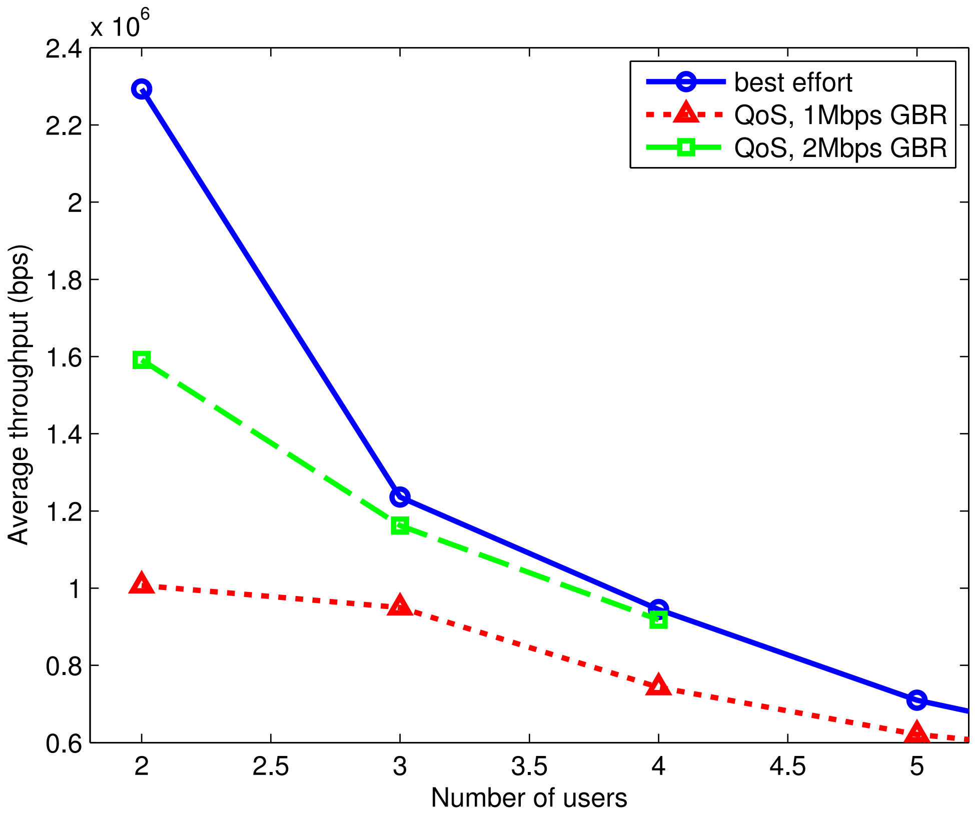


Figure 2.2: Comparison of average throughput per user for different types of services

providing the QoS-guaranteed services requires that the scheduler is biased in assigning resources to meet the minimum service quality to some users, e.g., the F_{QoS} and F_{delay} terms must change dynamically as discussed previously. This usually results in reduced network efficiency as resources sometimes must be allocated for users with bad channel conditions to maintain their targeted GBR's, thus lowering the overall cell's throughput. Additionally, this implies that an admission control policy must be implemented in the network as well to negotiate and grant QoS guarantees for each data session.

To demonstrate the throughput characteristics, the benefits and drawbacks of both service strategies in a mobile network, simple simulations using a LTE simulator (also to be described in detail in Section 3.5) were conducted. In summary, the LTE simulator [22] simulates the behaviors of a LTE cell where there are a number of mobile users moving randomly with realistic radio channel emulation. Three simulation cases were conducted where all users received best-effort services, QoS services with 1Mbps and 2 Mbps GBR respectively. The total number of users in each simulation varied between two to five users to represent different traffic load in the cell. Finally, each of these users was receiving data traffic from a TCP source, a constant bitrate source at 1Mbps and 2Mbps for the best-effort, QoS with 1Mbps and 2Mbps GBR simulation cases respectively.

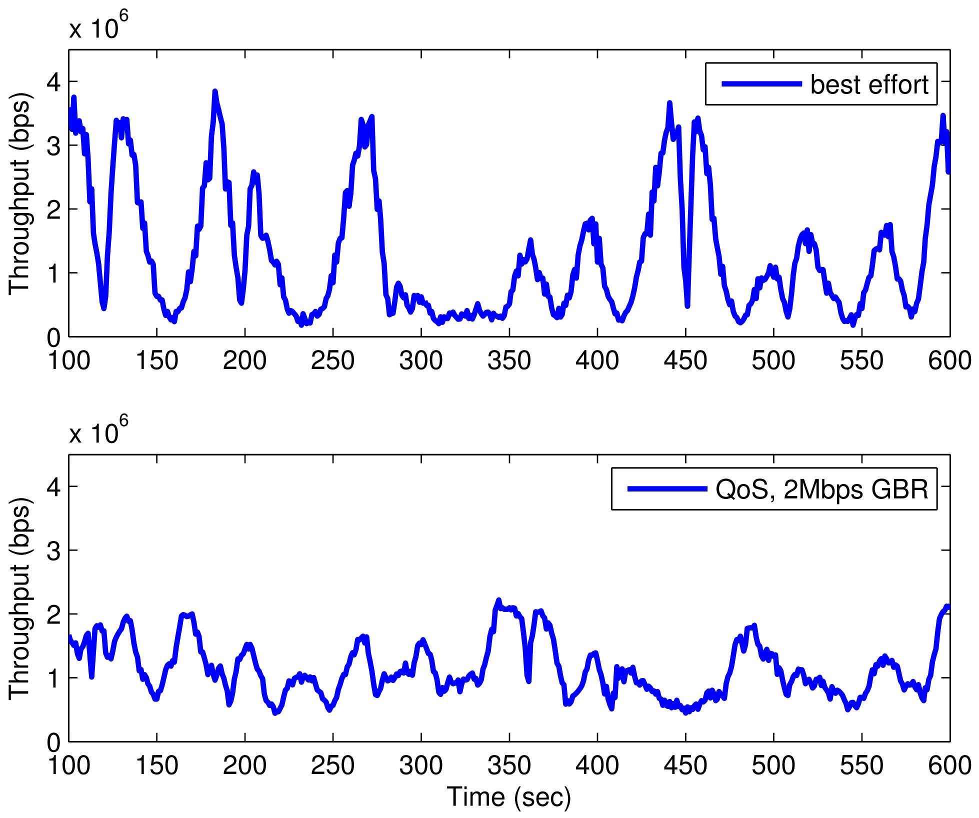


Figure 2.3: Comparison of instantaneous throughput for a best-effort and a QoS user

The average throughput per user in each simulation case is shown in Figure 2.2. For the QoS with 1Mbps GBR case, the cell is able to support only three or fewer users with this current cell configuration, but it becomes too congested and unable to maintain the GBR when there are four or more users in the cell. In the case of 2Mbps GBR, the cell appears to be too congested even with two users in the cell where the average bitrate is only 1.6Mbps. Note that if we compare the average bitrate between the best-effort and the QoS cases at the same number of users in the cell, it is found that the best-effort user always receives higher average throughput than the QoS one. This implies more efficient use of radio resources by the network which results in higher overall cell throughput as previously discussed.

Although there are many benefits to providing best-effort services from the efficiency point-of-view, providing QoS still offers some benefits in terms of reduction in bitrate fluctuation for the individual user. Figure 2.3 shows the instantaneous throughputs for a best-effort and a QoS user with 2Mbps GBR versus time where there are a total of three users in the cell. According to Figure 2.2, both the best-effort and the QoS users receive on average the same 1.2Mbps throughput. However, the instantaneous throughput for the QoS user is notably smoother and more stable than its best-effort counterpart where his instantaneous throughput swings between extreme high and low rapidly, depending on his reception quality. For example, the best-effort throughput reduced from 3Mbps to almost nothing at 275 seconds into the simulation and the outage continued for more than 60 seconds afterward. With such characteristics of the throughput, the adaptable bitrate range of the video content must be wider if the timestamp-based adaptive video streaming architecture is to be used with the best-effort service than the adaptable range required with the QoS service in order to avoid frequent playback interruptions. This implies the need to prepare more representations of the non-scalable video content or more scalable layers with the scalable video encoding (to be discussed in detail in Section 2.2) for the best-effort strategy. However, the throughput characteristic of the best-effort user is more suitable for video streaming with progressive downloading architecture since the user can exploit the brief moments with exceptional channel quality and large bitrate to request and cache a lot of video data well in advance. This is, however, not possible for the timestamp-based architecture where the transmission of packets is made based on their timestamps.

2.2 Preliminary on Video Coding and Quality Evaluation

Today's modern video encoding standards such as the H.26X family, MPEG1, MPEG2 and MPEG4 all share the similar concept to achieve data compression and are generally referred to as a block-based hybrid encoder [23]. The overall simplified encoding process for such encoder is depicted in Figure 2.4 and can be summarized as follows. First, each of the Y, Cb and Cr information of a picture is segmented into smaller rectangular areas, e.g., of size 16x16 samples. These are typically referred to as macroblocks. A set of prediction algorithms to find statistical correlation between blocks, either spatially or temporally or both, are performed. If the prediction is done temporally between pictures, this process is referred to as inter-picture prediction or motion compensated prediction (MCP) and the resulting prediction vectors are called motion vectors. Otherwise, it is referred to as intra-picture prediction. These prediction signals identify from which blocks the current one being encoded can be estimated. The difference between the prediction and the actual information in the block being encoded is likely to be close to zero or much smaller in value than its original signal. These differences in each block are then scaled and undergo frequency transformation, such as the Discrete Cosine Transform (DCT), to discard the few high-frequency components and the resulting transform coefficients are quantized. An entropy coding is performed on the quantized transform coefficients together with the motion information to further remove redundancy in the coded bitstream. Additionally, the encoder also reconstructs each encoded picture using the motion vectors and quantized transform coefficients it has just produced as well. This reconstructed picture, which is also the same as the decoded picture at the decoder, is used and stored for the intra/inter-

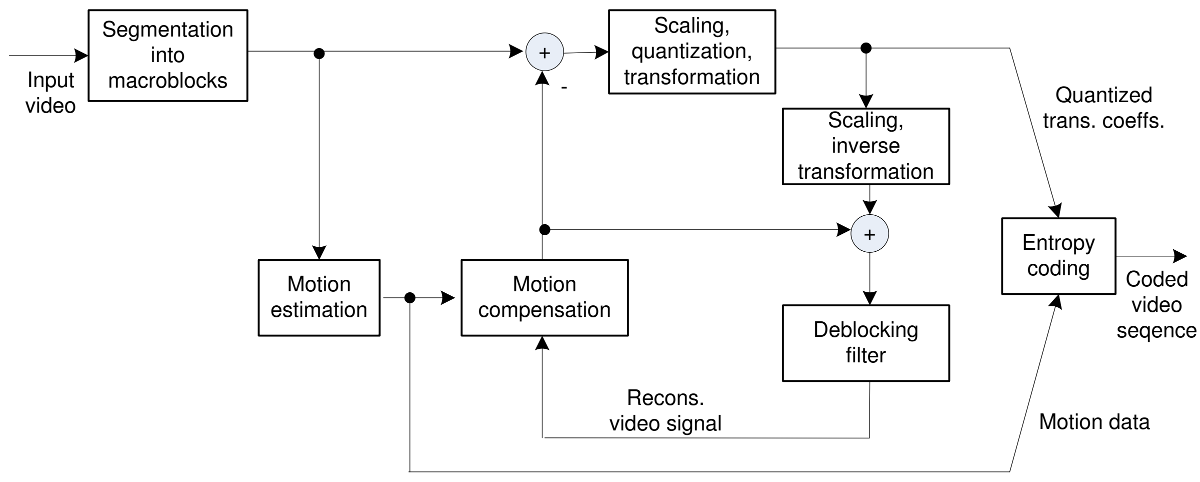


Figure 2.4: Generic digital video compression process

picture prediction process for the subsequent pictures.

To decode the compressed video bitstream, the decoder simply reverses the process done during the encoding. However, since the DCT and the quantization steps have already caused losses of some information, the decoded video can never be exactly the same as the raw video content.

2.2.1 H.264 Advanced Video Coding

The H.264 standard, also known as MPEG 4 part 10 Advanced Video Coding (AVC) [24], is the latest video compression standard jointly developed by the ITU-T Video Coding Experts Group (VCEG) and the ISO Moving Picture Experts Group (MPEG). It has been designed to be a high-efficiency encoding standard that is applicable to a wide range of applications; from a low-bitrate, low-latency mobile video conversation to high-bitrate, production quality usage. The encoding efficiency for the H.264/AVC has been significantly improved from its predecessors, e.g., the H.263 or the MPEG4 part 2, with bitrate savings of approximately 50% at the same level of visual quality. The H.264/AVC standard covers the two main relevant aspects, namely the Video Coding Layer (VCL) and the Network Abstraction Layer (NAL) which are briefly summarized as follows. More detailed information on the standard, the design concepts and its performance compared to others can be found in [23–25] and the references therein.

Video Coding Layer

This part of the standard involves details of the various techniques used for encoding and decoding of the video. The overall encoding process of H.264/AVC is still similar to what is shown in Figure 2.4. However, many restrictions due to the complexity concern in the older standards have been removed as well as some more complicated but efficient techniques are introduced as the hardware's capability improves. Some of the key aspects in the standard are as follows.

- Each picture is partitioned into rectangular macroblocks of size 16x16 samples for the Y component and 8x8 samples for the two chroma components. These are the basic building blocks on which the encoding is performed. The macroblocks are then grouped together to form slices, each of which can be independently decoded from others in the picture.
- Five fundamental slice types are defined. This includes the three types commonly found in previous standards, namely the I, P and B slices for which the macroblocks are only intra-picture coded, alternatively predicted from another picture and alternatively predicted from at most two other pictures respectively. The two additional slice types are the Switching P slice (SP) and the Switching I slice (SI) to assist in switching between different locations in the video or between videos without using the large I slice and for robustness against error propagation.
- The intra-picture coding for the Y component where the prediction is made from another area in the same picture can now be done with either a 4x4 prediction block for areas with fine details or a 16x16 prediction block for smooth areas with few details. However, the prediction block size for both chroma components is still 16x16 as in the previous standards.
- The inter-picture prediction for P slices is more flexible and accurate. The macroblocks in the P slice can be further partitioned into smaller blocks, e.g., 16x8, 8x16 and 8x8 samples for motion estimation which allows a finer cropping of moving objects of various sizes. The accuracy of the motion vectors is also increased to a quarter of a sample and can be arbitrarily weighted. Finally, although each block or macroblock in the P slice can have at most one motion vector that refers to just one previously decoded picture, different motion vectors for different blocks in the current picture do not have to refer to the same previous picture.
- The inter-picture prediction for B slices is similar to that of the P slices except that each block or macroblock in the current picture can have up to two motion vectors that refer to two different previously-decoded pictures and are combined using arbitrarily specified weights.
- The differences between the prediction signals and the actual values, also referred to as residuals, undergo discrete frequency transformation to represent them in terms of

a fewer frequency components instead. Then the transform coefficients are quantized with the desired Quantization Parameter (QP). However, H.264/AVC uses the integer transformation over a block of size 4x4 instead of the DCT over a block of size 8x8 typically used in the previous standards. Since the integer transform has an exact inverse function, inverse transformation mismatch and loss of information is avoided.

- The entropy coding, which reduces the statistical correlation between the quantized transform coefficients even further in the H.264/AVC standard, can be done with one of the two available algorithms; the Context Adaptive Variable-Length Coding (CAVLC) and the Context Adaptive Binary Arithmetic Coding (CABAC). Both of which are variable-length coding schemes that switch their coding tables based on the previous statistics, as opposed to schemes with one fixed coding table in the older standards. This results in a bitrate saving of approximately 10% to 15% at the same visual quality.
- The blocking artifact around the boundaries of the macroblocks is considered as one of the obvious artifacts, especially at a low bitrate, for a block-based video compression standard. The H.264/AVC standard has defined and included an in-loop deblocking filter in its decoder design to reduce the blockiness around these boundaries while still preserving the sharpness of real edges in the picture.

Network Abstraction Layer

This part of the standard involves preparing the coded video data for transmission through the underlying protocol layers. It defines the smallest unit of the packetized video data, referred to as the NAL unit, how they are arranged, segmented and the relationships between different NAL types in the video bitstream.

A NAL unit contains a one-Byte NAL header followed by an integer number of payload Bytes. NAL units can be classified into two broad types - the VCL NAL unit which contains the coded video data itself and the non-VCL NAL unit which contains side information relevant for decoding such as the Sequence Parameter Set NAL or the Picture Parameter Set NAL. Additionally, a group of NAL units with all relevant information to decode a single video picture is referred to as an access unit from now on.

Profiles and Levels

A profile is a set of supported compression techniques defined in the H.264/AVC standards that a decoder must be able to handle given a conforming video bitstream. A conforming encoder for a particular profile however is not required to utilize all the available techniques of that profile, but can choose to employ just a subset of them. Different profiles have been defined to address a wide range of different applications' requirements. Four of such profiles are briefly explained in the following to serve as examples.

- *Baseline profile.* This profile is intended for low-complexity, low-cost and low-delay applications, such as mobile video conversation, with additional robustness against loss and error propagation. The features that are supported in this profile, apart from other basic techniques, includes encoding macroblocks in I, P, SI and SP slices and the CAVLC entropy coding.
- *Main profile.* This profile targets applications that require standard-definition, medium to high encoding efficiency and good video quality while the decoder's complexity is not really an issue, such as a digital TV broadcasting service. Encoding of I, P and B slices, weighted combining of motion vectors and the more efficient CABAC entropy coding are supported.
- *Extended profile.* Intended as a profile for video streaming applications, it has the robustness features against the loss and allows rapid stream switching similar to the *Baseline* profile while having relatively higher encoding efficiency. All five slice types are supported as well as other techniques in the *Baseline* and the *Main* profiles except the CABAC entropy coding.
- *High profile.* This profile is designed for high-quality video streaming, broadcasting and storage applications that do not require too much protection and robustness from errors. The tools that are available to this profile are mainly to achieve maximum encoding efficiency. These are, e.g., usage of the I, P, B slices and the CABAC entropy coding.

Finally, a level defines operating ranges of some characteristics of the decoder and the encoded video bitstream. There are 15 different levels defined in the standard to imply different decoder's capabilities, e.g., the upper limit on the amount of macroblocks per picture, decoder's processing rate, the video bitrate, etc.

2.2.2 The Scalable Video Coding Extension of H.264

In 2007, the Joint Video Team of the ITU-T VCEQ and the ISO MPEG has released a new version of the standard with the Scalable Video Coding (SVC) extension in its Annex G. The SVC provides the ability to remove parts of the encoded video to reduce the bitrate while the remaining part still forms a valid video bitstream with a lower visual quality until the AVC-conforming base layer is reached. This graceful degradation property of the SVC is useful for various types of applications. This includes, e.g., a digital video broadcasting over a lossy satellite channel (DVB-SH) where different scalable layers receive unequal error protections to match the targeted coverage areas, video qualities and different receivers' capabilities [26, 27]. Another application that can benefit from SVC is the video streaming over a lossy and fluctuating mobile channel. In such a scenario, the ability to easily match the video bitrate to the available throughput in the mobile network and the terminal's capability without complex transcoding operations or switching between

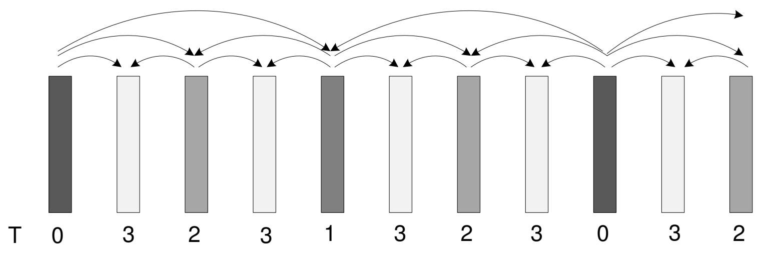


Figure 2.5: An example prediction structure for temporal scalability

multiple representations of the same content can potentially improve the users' perceived quality of service and the network's efficiency significantly.

There are three types of scalabilities supported by the SVC extension - the temporal scalability, the spatial scalability and the quality scalability. These are briefly explained in the following. A more detailed overview of the standard and how scalable layers can be optimally removed can be found in [3, 28] and the references therein.

Temporal Scalability

The temporal scalability refers to the ability to remove parts of the video bitstream and the remaining substream is still a valid video with a lower frame rate. Each temporal layer in the SVC context is identified by a temporal layer index T, starting from $T = 0$ for the base temporal layer with the lowest frame rate. The temporal scalability can be achieved by using the hierarchical prediction structure, which is to restrict the inter-picture prediction for P and B slices to make references to only other pictures with lower T indices than the current one. Thus, the decoding of each picture is independent of the other temporal enhancement pictures with higher T indices. Figure 2.5 shows an example of this concept where a video is hierarchically encoded with four temporal layers. The motion vectors in this example, represented by the solid arrow lines, all originate from frames with lower T indices than the one they predict. Note that a picture with a lower T index is usually encoded with a smaller QP to have a higher fidelity than a picture with a higher T index. This is due to the fact that the first one is used as a prediction reference for more pictures than the latter one.

Spatial Scalability

The spatial scalability refers to the ability to remove parts of the video bitstream such that the adapted substream still forms a valid video with a lower bitrate and spatial resolution. In the SVC context, each spatial resolution is identified by a dependency layer index D where $D = 0$ represents the base spatial layer with the lowest resolution of the video.

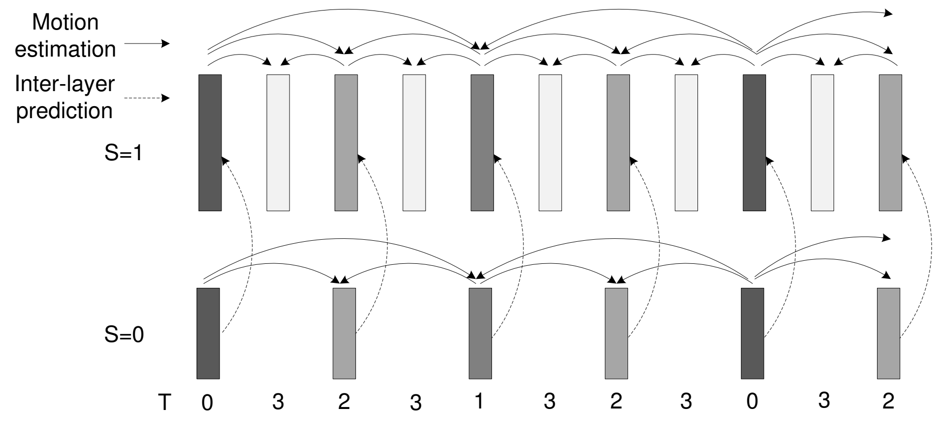
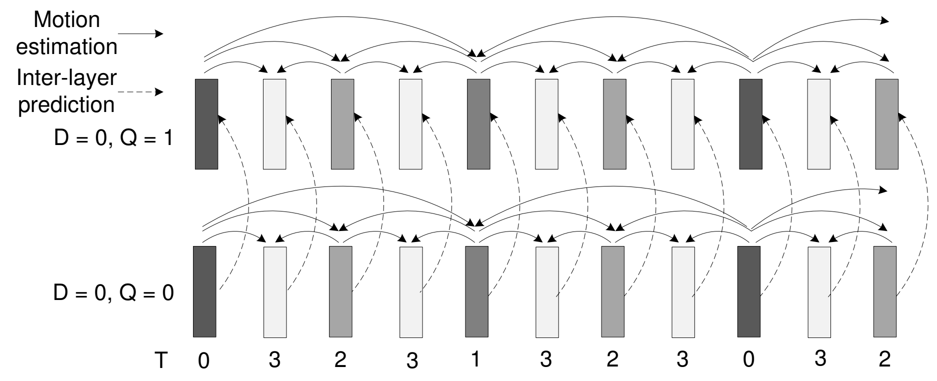


Figure 2.6: An example prediction structure for spatial scalability

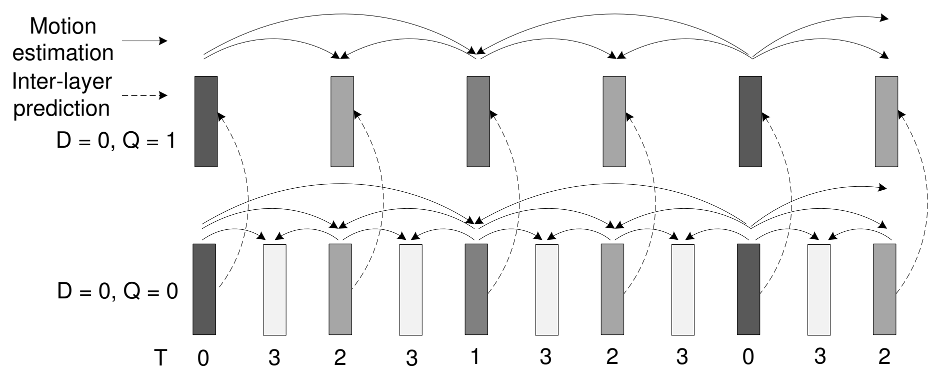
Within the same spatial layer, the two types of prediction technique, namely the intra-picture prediction and the inter-picture motion estimation, from the AVC standard are still used. However, to exploit as much information from the lower spatial layer as possible, a new inter-layer prediction is introduced which is basically an upsampled signal from the preceding lower spatial layer. A deblocking filter is then applied over the upsampled signal to minimize the blocking artifacts commonly found in such a process. The inter-layer prediction is constrained to be used only between spatial layers within the same access unit to reduce the design complexity. Since the upsampled information from the lower layer is not necessarily the best prediction signal for the current spatial layer, the encoder is allowed to choose whether to use the inter-layer prediction or the intra-layer motion estimation or to combine both, depending on the characteristics of the picture. Figure 2.6 demonstrates an example of a video with two spatial layers and the prediction vectors between pictures and layers. Note that this is also an example on combining spatial scalability with temporal scalability as well as the frame rate of the lower spatial layer is lower than that of the higher spatial layer.

Quality Scalability

The quality scalability, also referred to as the SNR scalability, allows for the extraction of substreams from the video with lower bitrates and picture qualities while maintaining the same spatial resolution. The SVC standard provides several means to encode videos with the quality scalability. One of which is to use the same techniques as in the spatial scalability case which is to use the inter-layer prediction between layers but with the same spatial resolution instead. In such case, the inter-layer prediction signals for the higher layers contain refinement information from using smaller QP values. This is referred to as the Coarse-Grain Scalability (CGS) and each quality layer is identified with the layer index D as before. However, switching between the CGS quality layers can be done at



(a) The completely intact video structure



(b) The video structure where NALs with $Q = 1$ and $T = 3$ are removed

Figure 2.7: An example prediction structure for two MGS scalability layers

only some specific access units, referred to as the Instantaneous Decoding Refresh (IDR) pictures where decoding does not require information from the preceding pictures.

To allow a greater flexibility for switching between the quality layers, the high-level signalling within the bitstream can be modified such that switching between quality layers can be done at any access unit as well. This is referred to as the Medium-Grain Scalability (MGS). Each MGS quality layer is identified by a new quality layer index Q where $Q = 0$ refers to the base quality layer. Thus, within the same dependency layer D, there can be more than one MGS quality layers Q. Note that this is a purely high-level signalling issue as the same techniques used for intra and inter-layer predictions for both the CGS and MGS scalabilities are the same. The difference is simply that the decoder does not switch between quality layers if they are assigned with different D indices, but can do so if they are assigned with the same D index but different Q indices.

Figure 2.7(a) depicts an example SVC video with two MGS layer. The removal of the NAL units for the quality layer can be done at any access unit, as demonstrated in the Figure 2.7(b) where the NAL of the $Q = 1$ layer for the pictures with the temporal layer $T = 3$

is removed. This removal strategy - to remove the MGS NALs from the highest T layer to the lowest T layer consecutively - can result in a slight fluctuation in the visual quality from one picture to the next. However, it provides several additional operating points for the video on top of the existing number of MGS layers, e.g., there are five operating points in this example even though there is only one MGS layer. This additionally reduces the amount of signaling overhead as well.

Figure 2.8(a) also depicts an example of an SVC video with three MGS layers. However, Figure 2.8(b) shows another alternative strategy to remove the NALs from the video which is to remove the entire Q layer completely each time. In this example, the quality layer $Q = 2$ is removed. The resulting video quality does not to fluctuate from one picture to another, unlike the previous removal strategy. However, it requires a large number of MGS layers to achieve the same amount of operating points in the video, thus creating more overhead and reducing the encoding efficiency as a result.

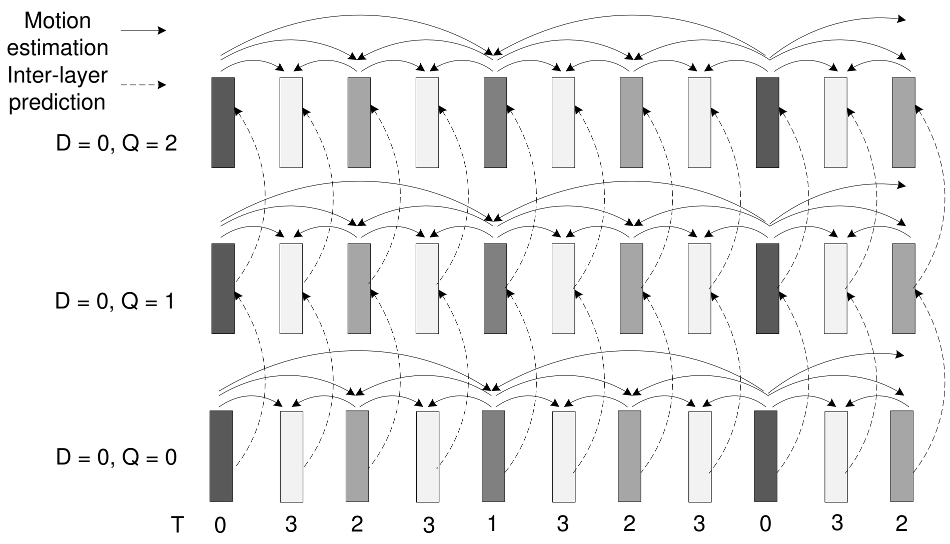
Network Abstraction Layer

The SVC extension still retains all the definitions and relationships between various NAL unit types of the H.264/AVC standard. However, the header of the NAL that contains coded information of the enhancement layer, referred to as the SVC NAL, is extended by three Bytes to provide information necessary for bitstream adaptation including the T, D and Q indices. In addition, the so-called prefix NAL unit which is a small NAL unit used to convey the T, D and Q information for a non-SVC NAL unit is introduced. This NAL type is to be placed before each of the non-SVC NAL units in the video.

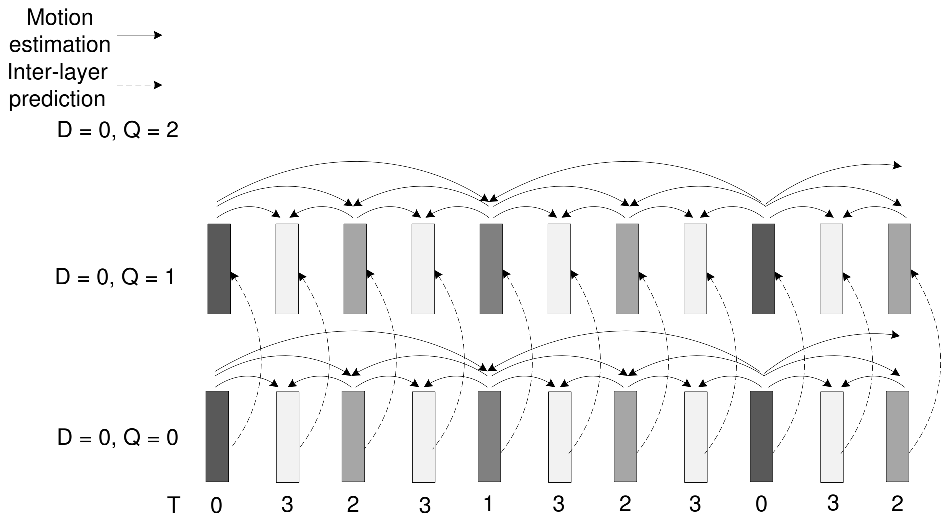
Profiles

The SVC extension defines three more additional profiles as follows.

- *Scalable Baseline profile.* This profile targets low-complexity and low-delay applications such as real-time video conversation and surveillance cameras. The AVC-compliance base layer of the video is required to conform with the original *Baseline* profile. The spatial layer, if exists, is restricted to have a resolution ratio between the higher layer to the lower layer of either 1.5 or 2. However, there is no restriction on the temporal and quality layers. Additionally, the use of B slices, weighted prediction and CABAC entropy coding are allowed in the enhancement layers although these tools are restricted in the original *Baseline* profile.
- *Scalable High profile.* This profile is designed for the same application types as the original *High* profile of H.264/AVC in which the base layer of the video needs to conform with such profile as well. There is no restriction on any of the scalability types supported by the SVC extension.



(a) The completely intact video structure



(b) The video structure where NALs with $Q = 2$ are removed

Figure 2.8: An example prediction structure for three MGS scalability layers

- *Scalable High Intra profile.* This profile is aimed only for professional uses which require high production quality of the video. This profile is similar to the *Scalable High* profile with the additional constraint that only the IDR pictures are allowed to be use both in the AVC base layer and all the scalable layers, effectively limiting the use of the inter-picture motion prediction in all scalable layers.

2.2.3 Key Performance Indicators

Video quality metrics to evaluate the quality of the reconstructed videos can be broadly categorized into two groups - subjective and objective quality metrics. The subjective video quality metrics involve collecting and processing individual quality ratings toward a video by a panel of viewers in a controlled environment. This is usually referred to as the Mean Opinion Score (MOS) which reflects the true users' perception and the level of satisfaction. Maximizing the MOS can therefore be considered as the ultimate objective of any adaptive video streaming architecture design. However, this type of quality metric is difficult to use, especially if there are a lot of videos from various simulations to evaluate, as it involves a lengthy process of subjective scoring by a panel of viewers. On the contrary, objective video quality metrics can usually be computed faster from the reconstructed videos directly. Some of these metrics simply compute the differences Byte by Byte to the original video material without taking the content into account. On the other hand, some more complicated objective quality metrics also consider the features of the video, the types of distortion and their effects to the human's perception as well. An overview of various available quality metrics along with related standardization bodies can be found in [29]. In this work, common Key Performance Indicators (KPIs) used in the later chapters are all objective video quality measurements. Although these KPIs do not directly reflect the actual human's perception on the video quality, considering them in combination and understanding their strengths and weaknesses still allow meaningful video quality comparisons to be made. These KPIs are as follows.

Peak Signal-to-Noise Ratio

The Peak Signal-to-Noise Ratio (PSNR) is a commonly used objective video quality metric due to its simplicity. By definition, it is a measure of the peak signal's strength to the error in the reconstructed video and can be considered as a logarithmic representation of the Mean Squared Error (MSE). For each frame of the video, the PSNR per frame, $PSNR_f$, can be computed as

$$PSNR_f = 10 \cdot \log \left[\frac{w \cdot h \cdot PeakSig^2}{\sum_{i=1}^{w} \sum_{j=1}^{h} (x_{i,j} - y_{i,j})^2} \right] \tag{2.2}$$

$$PSNR_f = 10 \cdot \log \left[\frac{PeakSig^2}{MSE_f} \right] \tag{2.3}$$

where $PeakSig$ is the largest value of each pixel which is 255 for a 8-bit representation, w and h is the width and height in pixel for each frame, $x_{i,j}$ and $y_{i,j}$ are the values of the original and reconstructed pixels respectively and MSE_f is the MSE per frame.

The time-averaged PSNR over the entire length of a video is computed from a logarithmic representation of the time-averaged MSE per frame. Let there be F frames in total, then the average PSNR can be written as follows.

$$PSNR = 10 \cdot \log \left[\frac{F \cdot PeakSig^2}{\sum_{f=1}^{F} MSE_f} \right] \tag{2.4}$$

PSNR is a good evaluation metric for reconstructed videos that might have been distorted during the transmission process, e.g., videos that have been re-encoded, have some scalable layers removed and/or corrupted by losses. However, the PSNR metric should only be considered as an approximation of the user's perceived video quality as it does not necessarily correlate with the viewers' opinions in all the cases. This is because the PSNR only compares the reconstructed video with the original one Byte-wise without taking into account different human's sensitivities toward different types of spatial and temporal distortions in the video [29]. This metric also does not represent the negative effects playback interruptions have on the user's satisfaction. Therefore, it alone is insufficient to provide a complete picture of a streaming architecture's overall performance, especially the one that is susceptible to an unbounded delay, e.g., progressive download video streaming with TCP. In addition, the PSNR metric needs to have the reconstructed and the original videos perfectly in synchrony frame-by-frame. If the reconstructed video has been temporally scaled down or affected by losses such that some frames could not be decoded, one must take care that the missing frames are inserted, e.g., by frame repetition, during the post-processing step before calculating the PSNR. This is to keep the number of frames in both videos equal and synchronous.

Pause Intensity

To quantify the severity of the interrupted playback events during the streaming, a simple metric that represents the relative amount of the total interruption duration to the original uninterrupted playback length, referred to as the pause intensity I_p, is defined as follows [30].

$$I_p = \frac{\text{total interruption duration}}{\text{original playback duration}} \tag{2.5}$$

The I_p ranges from zero in the ideal case to infinite. It is used together with other quality metrics to provide a better overall performance evaluation of an adaptive streaming architecture and as a basis for developing a more comprehensive quality metric to follow.

Video Quality Metric

The National Telecommunications and Information Administration (NTIA) has developed its General Model for video quality estimation, also referred to as the Video Quality Metric (VQM), as a perception-based objective video quality measurement. The metric has been designed and built from an extensive database of subjective tests to be able to accurately predict the perceived video quality for a wide range of video bitrates. It has also been independently evaluated by the Video Quality Experts Group (VQEG) and rated as one of the top-performing metrics in the tests. As a result, several standardization bodies have adopted the VQM as one of their normative quality metrics, e.g., the VQEG, the American National Standards Institute (ANSI) and the International Telecommunication Union (ITU) [31].

The VQM does not rely on the direct Byte-by-Byte comparison as used by the PSNR metric, instead it estimates the viewer's perceived quality by linearly combining several types of video quality distortions perceptible by the human visual system. These are referred to as the "quality parameters". Each quality parameter represents perceptible differences in a "quality feature" between the original and the reconstructed videos. A quality feature is a mathematical representation of a certain feature of interest in the video, e.g., edge information, chromatic distribution and the degree of movement of a selected Spatial-Temporal (S-T) region. The S-T regions are selected to represent only the portions of the screen that draw the attention of the viewer, e.g., the center portion excluding the area around the edges of the screen. A simplified summary of steps to obtain one of the quality parameters from both videos is as follows.

- In each S-T region, apply digital filters to enhance the feature of interest and calculate its representative value using simple mathematical operations, e.g., the mean and the standard deviation.
- A stream of feature values for all S-T regions in the video is obtained. These values can be further limited to be within a certain perceptible range of the human's visual system.
- Compare streams of feature values from the original and reconstructed videos with a suitable function, e.g., calculating their differences, a ratio or a logarithmic relationship between them. A stream of quality parameters for S-T regions is obtained as a result.
- The quality parameter values in the parameter stream are "collapsed", both spatially and temporally using a suitable method, e.g., calculating the mean, standard

deviation, taking the worst 5% values, etc. This is to reduce them to be a single real-number quality metric to represent such parameter instead.

The VQM uses a total of seven different quality parameters derived from seven quality features. These quality parameters are computed independently from the Y, C_b and C_r information and represent various types of visual impairments as follows.

- *si_loss* is the parameter that represents the spatial information losses, such as losses in the edge information introduced by blurring.
- *hv_loss* is the parameter that represents the relative losses of spatial information in the horizontal and vertical directions compared to those in the diagonal directions.
- *hv_gain* is the parameter that represents the unintentional increase of spatial information in the horizontal and vertical directions as a result of, e.g., blocking artifacts.
- *chroma_spread* detects the changes in the distribution of colors.
- *si_gain* represents the gain in the spatial information of the reconstructed video which could be the result from edge-sharpening measures employed by the video decoder.
- *ct_ati_gain* is the parameter that accounts for the masking effect of temporal impairments if there are a lot of spatial activities in the scene, causing them to be less perceivable. A similar masking effect of spatial impairments when there exist a lot of movements in the scene also applies and are taken into account.
- *chroma_extreme* detects severe localized distortion in color space, e.g., resulting artifacts from an error concealment where some blocks are replaced by a solid color.

Once these individual seven quality parameters have been computed, the VQM score is calculated by linearly combining them. The coefficients for the linear combination have been determined such that the correlation between the predicted VQM and the used database of subjective tests is maximized.

$$\begin{aligned} VQM = {} & -0.2097 \cdot si\_loss + 0.5969 \cdot hv\_loss \\ & + 0.2483 \cdot hv\_gain + 0.0192 \cdot chroma\_spread \\ & - 2.3416 \cdot si\_gain + 0.0431 \cdot ct\_ati\_gain \\ & + 0.0076 \cdot chroma\_extreme \end{aligned} \tag{2.6}$$

The resulting VQM score ranges from zero for the best quality with no distortion to approximately one in the worst case. This score can be scaled into any desired range, such as from one to five to represent a typical scale of the MOS score as well, where five is the best perceived quality. Note that it is possible for the VQM to be slightly higher than one if the reconstructed video is severely corrupted beyond the original database of subjective test results used to build the model. More details on the calculations of these quality parameters, the calibration process of the VQM metric and the verification results are described in [31–33]. In addition, the software to compute the VQM score given the original and reconstructed videos is also available from the ITS [34].

Given that the VQM has been adopted by various standardization bodies, its good performance in estimating the perceived video quality and the availability of the calculation software, it is therefore used as one of the KPIs in this thesis work as well. However, the resulting VQM score is further scaled up and inversed to take the value from one to five instead, representing the worst and the best subjective video quality ratings respectively which is a typical range for a MOS score. The equation to perform the scaling to obtain the estimated MOS, referred to as MOS_{VQM}, is as follows.

$$MOS_{VQM} = \max\left(5 - 4 \cdot VQM, 1\right) \tag{2.7}$$

In spite of its many benefits, the MOS_{VQM} still does not reflect the degradation of the perceived quality as a function of interruptions. It therefore must be used together with other KPIs in order to make a meaningful comparison between different adaptive streaming architectures.

Weighted Mean Opinion Score

Many studies such as [30,35–37] have revealed through various subjective tests that frequent and long interruptions can severely lower the user's perceived quality of service, e.g., the MOS_{VQM} discussed earlier. [35, 36] also suggest that the degradation to the MOS_{VQM} from the uninterrupted video can be modeled by a multiplication factor as a function of the pause intensity I_p. This new scaled MOS_{VQM}, referred to as the $MOS_{weighted}$ from now on, can then be obtained as follows.

$$MOS_{weighted} = \max\left(F_{MOS}\left(I_p\right) \cdot MOS_{VQM}, 1\right) \tag{2.8}$$

The $F_{MOS}\left(I_p\right)$ is a non-linear decreasing function of I_p and takes the value between zero and one to scale down the MOS_{VQM}. The $MOS_{weighted}$ is further limited to always be greater than or equal to one to stay within a typical range of a MOS. Figure 2.9 shows the resulting estimated $MOS_{weighted}$ by the $F_{MOS}\left(I_p\right)$ model in [36] which was derived based on subjective test results with low-bitrate, low-quality QCIF videos (from 32 to 256 kbps). The resulting $F_{MOS}\left(I_p\right)$ is described approximately by the following equation.

$$F_{MOS}\left(I_p\right) = \begin{cases} -1.71 \cdot I_p + 1.00, & I_p \leq 0.17 \\ -0.27 \cdot I_p + 0.75, & 0.17 \leq I_p \leq 2.79 \\ 0 & I_p > 2.79 \end{cases} \tag{2.9}$$

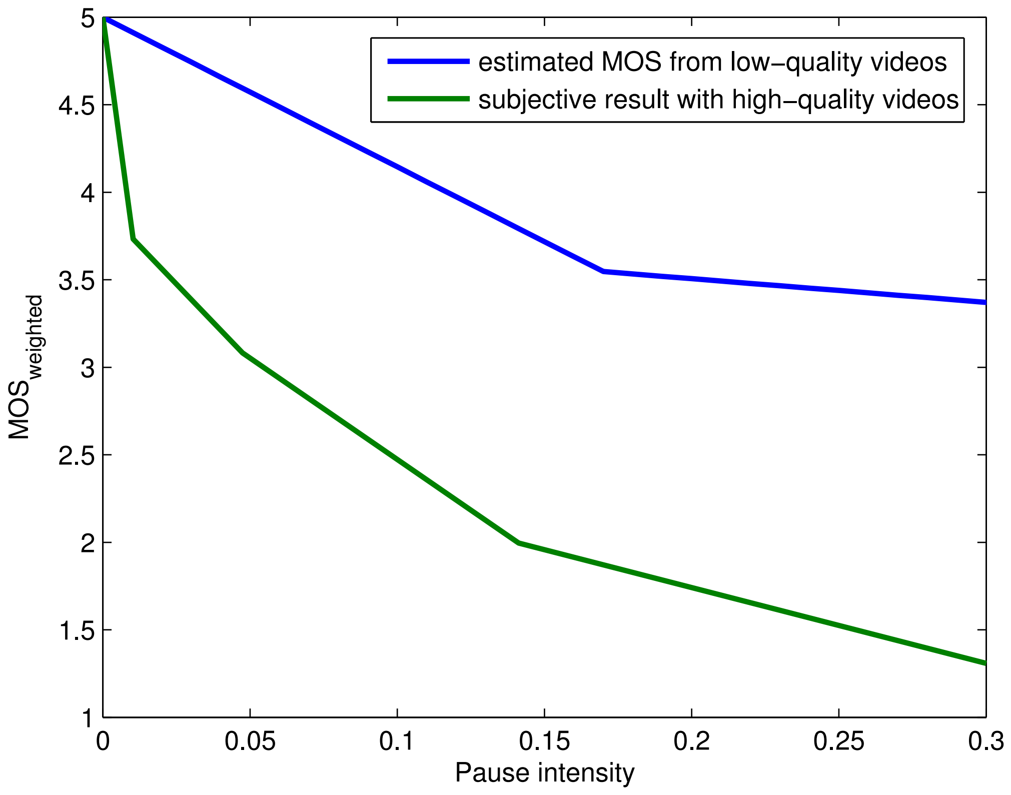


Figure 2.9: Degradation effect on the MOS from pause intensity (repreduced from [30,36])

However, the exact shape of the $F_{MOS}(I_p)$ also depends on the quality of the original videos used in the test as well. The subjective tests in [35, 36] suggest that videos with higher quality, e.g., larger bitrates and resolutions, tend to suffer more severe quality degradation from the interruptions than those with lower quality. This results in a steeper decline of $F_{MOS}(I_p)$ with increasing I_p for the high-quality videos. An example of the real MOS obtained from subjective tests with better quality videos in [30] which are rated at approximately 900 kbps is also shown in Figure 2.9 for comparison. This clearly reveals a more severe quality degradation effect with I_p compared to the $MOS_{weighted}$ obtained from (2.8) and (2.9).

In spite of the obvious dependency of the $F_{MOS}(I_p)$ to the quality of the original video, none of these works has provided a well-defined mathematical model to adjust the $F_{MOS}(I_p)$ accordingly. In addition, relationships between the perceived video quality and other buffering-related impairments, such as the initial pre-buffering length and the interruption locations, have been shown to exist but no concrete mathematical model has been developed for such relationships yet. Therefore, these are considered only preliminary works and require further studies in the future, especially given the growing popularity of video streaming with TCP where losses are converted into unbounded delay and more frequent interruptions.

In the later parts of this thesis work, videos with much higher quality and bitrates are used for simulations. Although there exists no precise mathematical relationship on how to modify the $F_{MOS}(I_p)$ in (2.9) to better suit with high-quality videos, it is sufficient to use the $F_{MOS}(I_p)$ as it is and consider the resulting $MOS_{weighted}$ to be an optimistic upper bound for the user's perceived quality. The $MOS_{weighted}$, combined with other previously discussed KPIs, can still be used together to provide a thorough performance evaluation in terms of the resulting video quality and a fair comparison between different video streaming architectures later on in this dissertation.

In addition to these introduced KPIs for video quality, there are also other alternatives to obtain the estimated MOS which can be used instead of the selected VQM in this work if desired. An example for such alternatives is the structural similarity metric (SSIM) as described in [38, 39] which gives an estimated MOS between zero (worst quality) to one (best quality) based on the structural distortions and information losses perceptible to the human's visual system in the reconstructed video compared to the original one. However, the VQM has been selected as the preferred quality metric of choice due to its wide acceptance, proven high correlation with the databases of real subjective tests and the availability of free calculation software. Finally, there are also other performance indicators used specifically for each of the proposed adaptive streaming architecture. Since they are used to demonstrate and compare only some certain aspects specific to the architectures, they will be discussed in detail later on in their respective chapters.

2.3 State-of-the-Art Video Streaming Architectures

Video streaming services can be broadly classified into two different paradigms which are referred to as the timestamp-based streaming (TBS) and progressive download (PD) throughout the rest of this thesis. This categorization is done based generally on whether the transmission of video packets from the server must adhere to certain deadlines or not, which ultimately influences the designs of the architectures and the supporting protocols. Different classifications of the video streaming architectures are possible. For example, [40] categorizes them into push-based and pull-based architectures, depending on whether the server actively "pushes" the video content through to the users or simply waits for requests first. Nevertheless, most of the push-based architectures also fall into the TBS category and similarly for the pull-based architectures with the PD category as well.

This section provides some preliminaries on both the TBS and PD streaming paradigms including their characteristics, advantages, disadvantages, supporting protocols, etc. Related works and proposed solutions to provide adaptive streaming for both paradigms as well as the current state of the art are also discussed.

2.3.1 Timestamp-Based Streaming

A timestamp-based video streaming architecture is a video delivery system over the Internet where the transmission rate from the server is similar to the encoding rate of the video, e.g., video packets are transmitted according to their encoding/decoding timestamps. Thus, the transmission rate of a TBS session rarely occupies the entire available network capacity in low load or good channel situations, but is limited to the bitrate of the video content itself. After a brief initial buffering period, the user starts to consume the buffered video data while his buffer is also being constantly replenished with new data from the server at approximately the same rate in an ideal situation.

Due to the fact that the TBS architecture does not pre-buffer a lot of video data too far in advance at the user, but relies on timely delivery of packets by the network, the suitable underlying transport protocols should emphasis more on minimizing the delay than providing error-free reliable delivery of packets. RTP/UDP [41] are therefore often the protocols of choice for such an architecture due to the absence of built-in retransmission and congestion control mechanisms that incur excessive delay. Retransmission of lost packets is usually left for the application to decide instead whether it wants to recover these packets or not. Additionally, TBS is usually a stateful architecture, that is the user needs to establish a session with the server first so that both of them are in the connected state before the the server can start transmitting video packets. This requires the assistance from other control protocols such as SDP, SIP and RTSP [42–44] to initiate the connection as well as RTCP [41] for regular exchange of network statistics and control information during the session.

Video contents to be used in the TBS architecture are usually prepared as single files to be transmitted atomically once connections have been established. The encoding structure of the video might also contain special frames that can be independently decoded from other frames regularly, e.g., the IDR frames, to support scrolling, fast-forwarding and rewinding features as well as increased robustness against losses. Nevertheless, each file still represents a very long duration or the entire video, since segmenting it into multiple fragments would require setting up a session for each of them during streaming, thus introducing unnecessary complexity and delay. This is a disadvantage considering that the video files cannot be easily distributed to various caching servers in the current Internet infrastructure. The need to have a unicast session directly between the origin server and each user in the case of on-demand streaming instead of reusing the existing caching servers means the origin server has to handle all the loads. This is likely to increase the congestion in the core network between the origin server and the access networks as well. Alternatively, there can be multiple representations at different bitrate and quality levels, or a single SVC-encoded file with multiple enhancement layers for each video content to provide some degrees of scalability. For a static adaptation to the user's capability, the server can provide a list of all available versions or scalable layers to the user to choose during session initialization. For a dynamic adaptation to the mobile channel, using SVC-encoded videos allows the server to trim the transmission rate by simply adding or removing enhancement layers on-the-fly without having to terminate the current session. Note that this is rather difficult with the non-scalable AVC encoding which usually requires setting up a new session every time the server decides to switch to another representation.

For the last-mile networks that are rather limited in terms of their throughput capacity, e.g., modem-based dial-up networks, 2.5G and early 3G, providing adequate throughput to support the stringent delay requirements of the TBS architecture proves to be a challenging task for most operators. This is also true even with the high-speed 3.5G or 4G networks in a congestion situation or when the user is in a bad reception area. Under these circumstances, the best-effort service in these mobile networks is often sporadic, unreliable and incurs too large delay variations for a continuous playback as shown earlier in Section 2.1.3. Thus, it is more desirable to use the QoS-guaranteed service for the TBS architecture in a mobile environment instead due to the ability of the user to negotiate for QoS supports from the network during session setup, e.g., the GBR and/or guaranteed delay. With the admission control mechanism in place, the network can reject new connection requests if they would jeopardize the QoS of existing TBS sessions to an unacceptable level. Additionally, Section 2.1.3 also shows that there is the benefit of having more stable instantaneous throughput to a mobile user by using the QoS service. This implies that the adaptable bitrate range of the video can be narrower, e.g., requiring fewer AVC representations or fewer scalable layers of SVC.

Traffic-Curve Analysis

In this section, the relationships between the delay, losses and interruptions during streaming for a typical TBS session are studied using traffic curve analysis and network calculus [45]. Consider a typical TBS session with no retransmission for lost packets and the following assumptions. If losses occur due to buffer overflow at some of the network nodes along the transmission path, the decoder will try to conceal the errors and continue on with the decoding as long as there are more packets in the user's receiving buffer to decode. Playback interruptions can occur only when the user's buffer is empty, but not because it is waiting for retransmissions of missing packets.

Define $B_{Tx}\left(t\right)$ as the accumulated amount of Bytes transmitted by the server and $B_{Rx}\left(t\right)$ as the amount of Bytes the user has received versus time. $B_D\left(t\right)$ is the total amount of Bytes that have been decoded and $B_L\left(t\right)$ is the total lost Bytes due to congestion up to time t. An example of these curves is depicted in Figure 2.10 where the server starts sending packets at $t = 0$ and the user starts receiving them after T_N seconds, representing the core network delay. Assuming the core network is always over-provisioned, T_N can be considered as a small positive constant. The shape of $B_{Rx}\left(t\right)$ is not necessarily identical to that of $B_{Tx}\left(t\right)$ due to additional variable queuing delay at the base station, depending on the instantaneous channel and the congestion level in the cell. However, $B_{Rx}\left(t\right)$ is bounded to be within a certain range. In an ideal case where the base station's queuing delay is

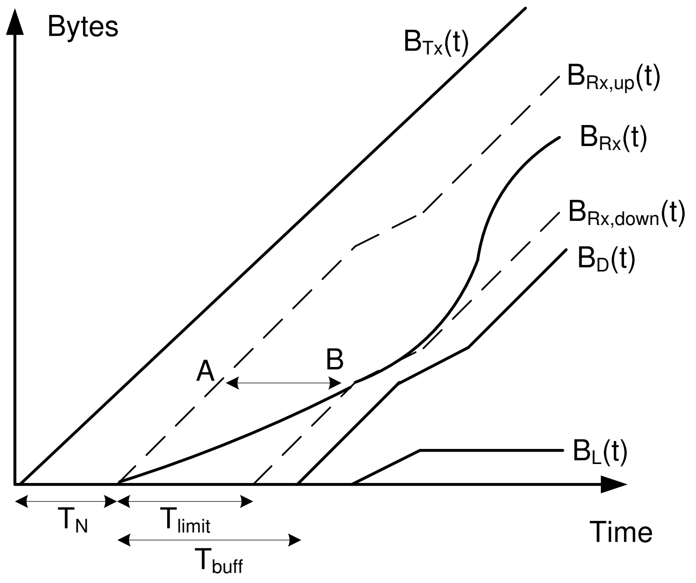


Figure 2.10: Example traffic curves for a timestamp-based streaming service

zero, an upper bound for $B_{Rx}(t)$, denoted by $B_{Rx,up}(t)$ can be derived by shifting $B_{Tx}(t)$ by T_N seconds and properly offsetting it with the losses as follows.

$$B_{Rx,up}(t) = B_{Tx}(t - T_N) - B_L(t) \tag{2.10}$$

Similarly, a lower bound $B_{Rx,down}(t)$ can also be derived in the same way but with the queuing delay equal to T_{limit} which represents a delay threshold of the base station before it starts dropping packets due to congestion.

$$B_{Rx,down}(t) = B_{Tx}(t - T_N - T_{limit}) - B_L(t) \tag{2.11}$$

As an example, the queuing delay at the point B in Figure 2.10, represented by the horizontal distance $\bar{AB}$ between the $B_{Rx}(t)$ and the $B_{Rx,up}(t)$, reaches the T_{limit}. The base station at this point starts to drop packets, causing the loss curve $B_L(t)$ to grow.

After an initial buffering period from receiving the first packet, denoted by T_{buff}, the user starts to decode the buffered data. The shape of the decoding curve $B_D(t)$ is similar to a shifted version of the $B_{Tx}(t)$ and properly offset by the losses, depending on where the losses hit the original video. If newly arrived packets are dropped first every time the buffer overflows, $B_D(t)$ is similar to a shifted version of the $B_{Rx,up}(t)$, assuming $T_{buff} \geq T_{limit}$ (since the $B_{Rx,up}(t)$ itself is already a shifted version of the $B_{Tx}(t)$ and offset by $B_L(t)$).

$$B_D(t) = B_{Rx,up}(t - T_{buff}). \tag{2.12}$$

Alternatively, if the Head-of-Line packets with the oldest queuing delay are dropped first as is the case in Fig. 2.10, we have

$$B_D\left(t\right) = B_{Rx,down}\left(t - \left(T_{buff} - T_{limit}\right)\right). \tag{2.13}$$

In such case, it can be shown that in theory $B_D\left(t\right)$ will not touch $B_{Rx,down}\left(t\right)$ during the entire streaming session as they are a shifted version of each other. In other words, the playback is unlikely to stall given that the initial buffering time is large enough, e.g., $T_{buff} \geq T_{limit}$. This finding applies regardless of whether the adaptation is done to the video or not, and therefore the reduction in interruption time is not expected to be a major benefit of any timestamp-based adaptive streaming architecture. Instead, the benefits of the adaptation are in terms of having fewer congestion-related losses and improved video quality compared to the non-adaptive one.

Related Works and the State of the Art

Providing QoS support and adaptation to video streaming services, especially the TBS architecture, has been of interest to the research community for many years. A comprehensive survey of various approaches provided in [46] categorizes them into two general groups - the network-centric approach and the end-system centric approach. The network-centric one, as the name suggests, concerns mostly on how the network can provide QoS support and adapt to the application's requirements, while the end-system centric one concerns the approaches for which the server and/or the user can adapt themselves to varying network conditions. The following literatures as well as the proposed adaptive TBS architecture to be discussed in Chapter 3 fall into the latter category as well.

Chou and Miao in [47] propose a generic way to design a Rate-Distortion (RD) optimized adaptive streaming architecture by formulating it into an optimization problem, taking into account relationships between parts of the video, the costs of transmitting them and the expected reduction in distortion each part contributes. An accurate statistical model of the channel is needed to model a probability of successful transmission over a mobile channel depending on reception quality, congestion level and other influential factors. Further adjustments to the statistical model are necessary to incorporate the congestion control and to coordinately adapt the videos for multiple users simultaneously which add more complexity to it. The adaptation results depend largely on how accurate the model represents the channel. Other works such as [48–52] opt for a simpler approach by assuming a constant limited amount of radio resources in which bitrates of all different users must be optimized together within this static resource budget instead. Thus, the accurate statistical model of the channel is not required. These works propose a joint optimization of the video bitrates by a dedicated adaptation entity based on various metrics. In [48–50], utility functions that model the estimated Quality of Experience (QoE) for different application types at various bitrates are used to assist the adaptation. The considered adaptation methods in some of the works are for non-scalable videos such as re-encoding and frame dropping. Thus the computational resource constraint, especially for those that perform re-encoding, needs to be taken into account as well. [48] additionally attempts to reduce fluctuations in the video quality as a result from adaptation by defining a threshold of human perception, then applying this additional constraint to the optimization process. [51] proposes a similar dedicated adaptation proxy close to or at the base station using cross-layer information, specifically the Link layer throughput and the current queue length, and the structure of the video for both coordinated and uncoordinated adaptation. A notable observation from these works is that a congestion control mechanism to regulate amount of available resources is not considered for the video streaming or any particular class of traffic. Instead, the amount of the available resource budget as an optimization constraint is the fixed total system resources. In [48–50], these are shared among all the users and application types within the cell while [51] assumes all users in the cell are video streaming users. Thus, detailed knowledges on, e.g., the number of users of each application type in the cell, their traffic characteristics, QoS requirements and a specific utility function for each of them are required to jointly adapt their bitrates. This increases the complexity of the optimization process and requires a lot of cross-layer information exchange between the base station and the optimization entity such that both are likely to be co-located and must be owned and operated by the network operator itself.

An alternative approach to having a centralized adaptation entity is to decompose the Network Utility Maximization (NUM) problem such that smaller subproblems can be formed and distributed to various network entities instead [53, 54]. This concept is applied to the problem of enhancing video streaming quality over a mobile network in, e.g., [52, 55, 56] where the decomposition of the NUM problem results in a distributed adaptive algorithm between the base station and the video servers or the mobile users. In [52,56], each streaming user determines its optimum amount of required resources given the "price" set by the base station and informs the latter of its decision while the base station has to constantly adjust the price based on the "demand". This process is done iteratively until the total required resources is within the resource budget, after which the base station further determines appropriate transmission schedules for video units to meet their decoding deadlines. The algorithm in [55] is similar but the iteration is done between the base station and the video servers themselves. Layering as decomposition optimization of the NUM is generally a good theoretical approach of getting mathematical insights into various cross-layer designs of a communication network, especially if one has the freedom to modify and re-allocate functionalities to different layers / network elements to optimally suit specific purposes. There are, however, some practical issues that can discourage the deployment of these works although the proposed solution eliminates the need for a centralized adaptation entity. Firstly, it still requires the base station to perform a centralized role of adjusting the resource price iteratively with either the servers or the users. This implies that the base stations are not just transparent nodes along the transmission path, but their IP addresses and other sensitive information, e.g., the user's channel quality indicator and the resource price (reflecting the relative user's location and traffic load in the cell respectively) must be made known to the video servers and/or the users. Also, the need to perform iterative computation between these entities could introduce additional delay, especially in a congested mobile cell. This could be remedied by having a dedicated high-priority control channel for exchanging messages between the base station and the involving parties at each iteration at the expense of increased complexity. For these reasons, the most practical deployment scenario to minimize the delay and the risks of exposing sensitive information to the outside is for the mobile network operator to own and operate the video servers itself. This however would result in limited variety of contents from other external video hosting and service providers.

The Datagram Congestion Control Protocol (DCCP) [57] is a recently-standardized transport protocol to be an alternative from the UDP for providing unreliable packet delivery with several built-in congestion control algorithms that the application can select from. One of which is the TCP-Friendly Rate Control (TFRC) [58, 59] which provides a relatively more stable throughput for video streaming applications than TCP while being fair to other competing TCP flows at the same time. The rate adaptation is done by the server for each individual streaming user using a rate control equation that estimates the TCP throughput under the same loss and delay situation. However, this state-of-the-art protocol can introduce excessive delay when the underlying TFRC algorithm refuses to allow sending packets at higher rate than what it deems appropriate. The transmission rate from the server is therefore not strictly timestamp-based and the traffic curve analysis made earlier does not apply. Since the fairness between traffic flows in a mobile cell is partially or wholly controlled by the base station's resource scheduler already and the application should try to utilize all the available radio resources given to it instead. DCCP protocol with TFRC can therefore be slightly too conservative which, in addition to its inability to perform coordinated adaptation, results in more frequent playback interruptions and lower overall video quality when used in a mobile environment as will be shown later on in Chapter 3.

Another end-system centric approach is to control when and how fast the buffered video data is consumed at the end user. A thorough summary of various playback strategies and related works can be found in [60]. Some examples of these are, e.g., [61, 62]. The first work models the resulting video quality in terms of various network characteristics and uses it to adjust the initial buffering time. While having a closed-form mathematical model of the video quality as a function of the network QoS is interesting, only modifying the initial buffering delay cannot guarantee good playback quality in the dynamic mobile environment. In addition, the algorithm needs to learn the channel characteristics at the beginning before the streaming can start, unavoidably adding additional delay. In [62], the authors propose content-aware playback and packet scheduling algorithms to minimize the effect of varying network throughput. An algorithm to adjust the playback speed at the user based on the buffer occupancy and motion intensity in the scene has been developed. The frame rate can be either slowed down or speeded up as necessary to avoid a complete stall from buffer underrun. A content-aware scheduling algorithm has also been introduced at the base station to discard video units that are too late to meet their decoding targets.

Some of the works, e.g., [63, 64] propose coordination between different protocol layers to better support mobile video streaming services. This could involve overriding internal operations of different layers, e.g., having the application influencing the lower layers in order to apply different Forward Error Correction (FEC) protection levels based on the importance of different video frames or other high-level network statistics. Yoshimura et al. propose a similar concept in [65] where an RTP monitoring agent is proposed to be located close to a wireless link. This agent sends RTCP reports on the congestion and loss statistics back to the server simultaneously with the mobile user itself. Thus, the server can distinguish between congestion-related losses in the core network and channel-related losses in the wireless link and take appropriate adaptation actions, e.g., whether to reduce the transmission rate or to increase the FEC strength for the packets. However, modern 3G/4G technologies all have built-in retransmission mechanisms at the Medium Access Control (MAC) layer to convert channel losses into congestion-related losses already. Thus the concept of having the server distinguishing different causes of losses is not applicable to the modern mobile networks anymore, let alone the fact that the selection of the FEC protection level for the radio channel is usually not the responsibility of the media server. In addition, the difficulties in having non-standardized cross-layer interfaces between different layers and network nodes and the additional complexity it involves often prevent this type of approach from real deployment.

Note that many of the proposed solutions require exchanging of side information between network entities to some certain extent, e.g., the RD information of the video, network statistics, resource budget, decoding deadlines, etc. While this is assumed to be implemented proprietarily in most works, it is worth noting that there exists the MPEG-21 Digital Item Adaptation (DIA) standard [66] as well which defines tools and metadata containers that can be used for such purpose. The Bitstream Syntax Description (BSD) [67] and the Usage Environment Description (UED), for example, define formats for XML-based tools that can be used to convey RD information of a video and dynamic network statistics between network entities respectively which can be extended to carry additional proprietary information as well. [68] also provides a conceptual adaptation architecture using these tools as an example how they can be utilized.

2.3.2 Progressive Download

For an access network that is capable of providing large and sustained best-effort throughput to mobile users, the TBS architecture is not an ideal method to provide video streaming service since the transmission rate from the server is limited to be only as high as the encoding bitrate. In such case, the additional capacity that the network is able to support is therefore underutilized. To some extent, this also applies for on-demand video streaming too as there is little need for the server to restrict its transmission rate to only the encoding bitrate or the negotiated GBR. Thus, the progressive download (PD) architecture which allows the server and users to ramp up the transmission rate to the network's capacity would be more appropriate for such scenario.

A user-driven video streaming architecture based on the PD concept, alternatively referred to as the pull-based architecture in [40], is a stateless streaming architecture. The streaming user makes the decision to request for parts or the entire video from the server while the server itself only waits and replies to the requests without needing to maintain a dedicate communication session between itself and the user. Since there is no limit on the maximum transmission rate, the supporting protocol needs to provide some means of congestion control to ensure fairness and stability of the network. Additionally, the user is likely to have enough time for retransmission of lost packets as well as it is allowed to buffer a lot of video data in advance. Given these requirements, HTTP/TCP [69] which provide stateless request-response mechanisms between the user and the server, reliable transmission and congestion control are almost always the protocols of choice. Note that the less-popular DCCP which provides congestion control mechanisms but not reliable transmission can potentially be used as the transport protocol for the PD as well. However, it is not compatible for the HTTP to be used on top since the latter protocol requires the lossless transmission service to function properly. The streaming application in this case therefore has to take responsibility for these missing functionalities by itself, unnecessarily complicating the design and implementation.

Since the PD is a stateless request-response architecture with no control protocol such as the RTSP to support features like scrolling and skipping, the video itself is usually prepared as multiple independent fragments, referred to as video chunks from now on, to support rapid playback on any part of the video without transmitting everything from the beginning again. Each video chunk is self-decodable and represents only a short period of the video, e.g., a collection of Group-of-Pictures (GoP) structures lasting for a few seconds with no dependency on other chunks. Note that segmenting the video into multiple smaller chunks is likely to reduce the encoding efficiency compared to having a single file as prediction vectors across chunk boundaries are not allowed. Furthermore, given the larger overhead of HTTP/TCP than RTP/UDP protocols, the PD architecture tends to be slightly inferior to the TBS one in this regard. However, having multiple video chunks allows them to be stored at various HTTP caching servers distributed all over the Internet which can then be used as additional video streaming servers without any modifications. The load to the origin server, congestion in the core network and transmission delay are therefore significantly reduced with content caching. Additionally, the PD architecture also has fewer problems with firewalls and Network Address Translation (NAT) as the transmission is done over HTTP/TCP just as typical web browsing traffic. These advantages make this architecture easier to be deployed than the TBS.

Traffic-Curve Analysis

This section investigates the relationships between different traffic curves of the PD architecture, similar to those for the TBS one introduced earlier. Consider a situation where

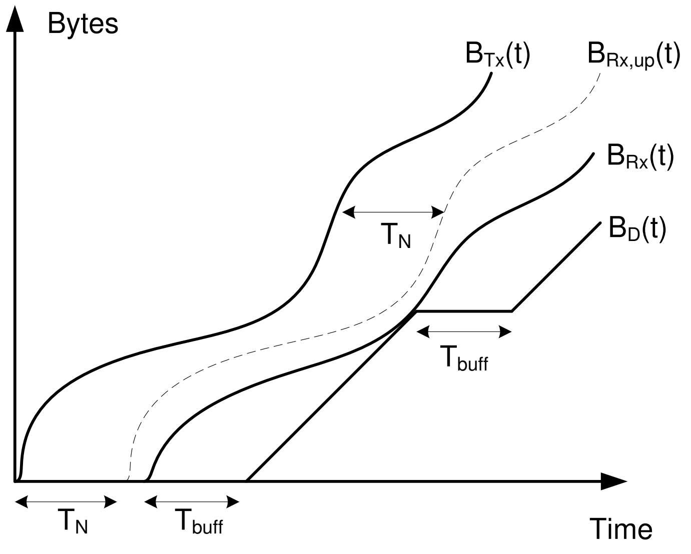


Figure 2.11: Example traffic curves for a non-adaptive progressive-downloading streaming service

a PD video streaming user is consuming a video content from a server. The transmission curve from the server, $B_{Tx}(t)$, can be any non-decreasing function whose slope reflects the available best-effort throughput. It is, however, not necessarily similar to the decoding curve, $B_D(t)$, whose slope is the encoding rate of the video itself. Also, since the transmission is done over HTTP/TCP, reliable packet delivery is guaranteed and the loss curve, $B_L(t)$, does not exist. Assuming the core network is again over-provisioned, the network delay, T_N, is a small positive constant. The upper bound of the receiving curve, $B_{Rx,up}(t)$, which represents the earliest arrival curve of the video data given no queuing delay at the base station at all is simply a shifted version of the $B_{Tx}(t)$ and can be written as follows.

$$B_{Rx,up}(t) = B_{Tx}(t - T_N) \tag{2.14}$$

However, the retransmission mechanism of the TCP where lost packets can be retransmitted indefinitely until they are all correctly received implies unlimited maximum packet delay, making it impossible to set a certain arrival deadline. In addition, the transmission rate from the server which is controlled by the TCP's congestion control algorithm can also be significantly lower than the encoding/decoding rate of the video as well if the congestion situation forces the TCP to lower the throughput. Strict transmission deadlines for packets therefore cannot be made. As a consequence, $B_{Rx}(t)$ cannot be lower-bounded. The implication that follows is that no matter how large the initial buffering period, T_{buff}, is for the PD paradigm, it is still impossible to guarantee smooth interruption-free playback. An example of such situation is shown in Figure 2.11 where the decoding curve, $B_D(t)$, which started T_{buff} seconds after the first packet had arrived, suffered buffer underrun later on

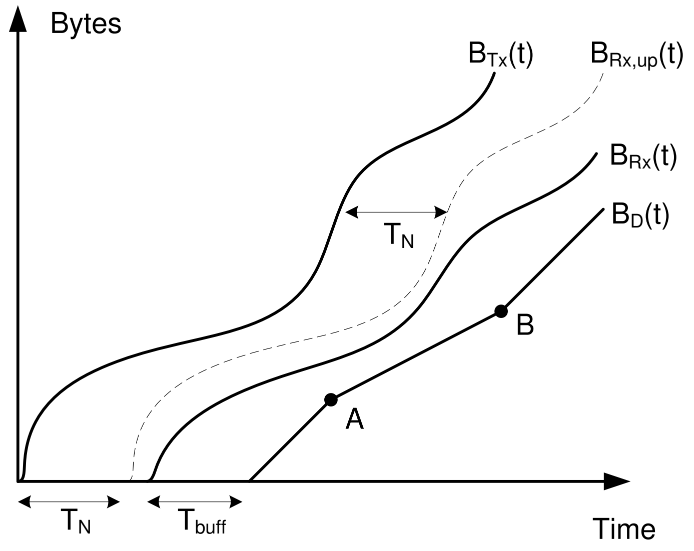


Figure 2.12: Example traffic curves for an adaptive progressive-downloading streaming service

and the playback was interrupted to wait for more data.

Although uninterrupted playback cannot be guaranteed for the PD architecture, the streaming user can still minimize or avoid it to some extent if dynamic adaptation to the channel is done. Consider the same situation but with the streaming user being able to adapt the video bitrate by, e.g., switching to a lower representation of the video as shown in Figure 2.12. Once increasing congestion was detected at point A, the user decided to request for a lower-bitrate version as seen here with $B_D\left(t\right)$ having more gradual slope between point A and B, and thus avoiding the potential interruption. Note that although the frequency and duration of interruptions can be reduced this way, the quality of the video is also worsen as a trade off.

Related Works and the State of the Art

Presently, the most prominent standardization effort for the PD architecture is the Dynamic Adaptive Streaming over HTTP (DASH) [9] which defines how the video contents are divided into smaller chunks (referred to as segments and subsegments by the standard) so that they can be stored at any typical HTTP servers, how the users can obtain metadata information and retrieve the individual parts of the videos. Further integration of the standard into the HTML5 [70] is also envisaged which will enable browsers with HTML5 support in the future to be able to consume DASH contents easily.

Providing adaptation capability for the PD architecture has mostly been done by having multiple AVC representations of the same content available at the server. The streaming user can perform both static and dynamic adaptations to match the video quality to its own capability at the beginning and to the current network conditions during streaming respectively. Unlike the TBS architecture, this is relatively easier with the PD paradigm as there is no need to terminate and set up a new session every time the user desires to switch to other representations when the transmission is done over HTTP (compared to a more complicated procedure of tearing down and setting up new RTP sessions). Since the DASH standard does not define the adaptation algorithm but leaves it open for different implementations, there have been several proposals for standard-compliance generic adaptation engines both from the research community and the industry, e.g, Microsoft's Smooth Streaming, Netflix and Apple's HTTP Live Streaming [71–76]. Akhshabi et al. [71] provide preliminary performance evaluation and comparison results between some of these commercially available algorithms in which they are all found to be able to cope with congestion scenarios in wired networks reasonably well. In [75], Liu et al. additionally propose having parallel HTTP connections to request for multiple chunks while also performing basic rate adaptation simultaneously. Having parallel HTTP connections has some additional benefits, e.g., increased utilization rate of the core network where different HTTP connections might be made to different caching servers. Additionally, it mitigates the effects a single HTTP connection might have on the overall throughput from occasional stalls in TCP throughput when it encounters losses or spikes in RTT delay. However, the basic concept of these adaptive algorithms are simply to select the representation for the next few chunks such that their required bitrates is no larger than the past average bitrate. A study by Ramshankar [77] on the performance of one of these algorithms [74] in a mobile environment reveals less-than-satisfactory results and that there is still much room for further improvements.

The usage of H.264/SVC with the PD architecture has also been of interest recently. A usual way to prepare a SVC-encoded video for this purpose and to be compliance with the DASH standard is to extract each scalable layer within each chunk into a separated file of its own, referred to as a block from now on. Thus, a chunk consists of several smaller blocks of scalable layers within it. Although a chunk can be independently requested and decoded from other adjacent chunks, the extracted layers from the same chunk are not (more details are covered later on in Chapter 4). Some obvious benefits to using SVC for static adaptation to heterogeneous devices, as demonstrated in [73, 78], include improved caching efficiency at the edge servers, reduced core network congestion and ease of content preparation by encoding the video only once with SVC. For dynamic adaptation to the varying channel, earlier works such as [79,80] propose a server-based adaptation mechanism to adapt the number of enhancement layers in each successive frame by means of observing TCP throughput and periodic reports of the playback buffer level from the user. A more recent work by Kuschnig et al. [81] proposes three different adaptive algorithms, two of which simply select the number of layers so that the next GoP has bitrate no larger than the past average bitrate. The third proposed algorithm can actually be classified as a TBS architecture, although it uses the TCP as the transport protocol. This is because the server in this proposal transmits GoP's based on their decoding timestamps. If the estimated time to send the next GoP through the TCP is longer than a GoP period, the server then reduces the number of layers in that GoP accordingly so that it can be sent within its deadline. These proposed adaptation algorithms are all server-driven which is not compatible with the newer DASH standard. The need to have the server making adaptation decision also compromises the scalability of the architecture to future network expansion as well. Additionally, most of the available dynamic adaptive algorithms to date, both for multiple AVC representations and SVC, simply select the next layer/representation to request by comparing the delay and/or the past average throughput with the bitrates of the available layers/versions. The next few chunks are then requested sequentially at the comparable bitrate. In case of SVC, they also do not permit requesting for separated enhancement layers, when opportunities allow, to upgrade already-received chunks which have not been decoded yet. This further limits the adaptation options and the ability to exploit the full potential of SVC. Finally, none of them, to the best of my knowledge at the time of writing, has directly address the rapid fluctuation of the best-effort throughput in a mobile environment and proposed a solution that also utilizes the SVC before.

The generic adaptation algorithm proposed by Chou and Miao [47] to perform RD-optimized adaptation can potentially be used in this scenario, but with modifications to the original concept. This is because the algorithm has originally been designed for video streaming over a lossy network, e.g., the TBS architecture with RTP/UDP, where the cost of transmitting a data unit is derived from, e.g., the level of FEC protection and the possible number of retransmission attempts. However, these are the responsibilities of the underlying MAC and Physical layers at the base station and the TCP layer at the server. The user's streaming application usually have no access to these layers/entities nor the knowledge of these informations. Instead, the cost of transmission should be represented in terms of a success probability in delivering the data units before their useful deadlines which is the approach taken in this work (to be discussed in detail in Chapter 4).

Chapter 3

Timestamp-Based Adaptive Video Streaming

3.1 Motivation

As pointed out earlier in Chapter 2, there has not been a large-scale commercial deployment of an adaptive TBS to date yet, let alone the ones specially designed for a mobile environment. Some of the obstacles, amongst other reasons, are the complexity of the proposed architectures and the requirement to have cross-layer communications and optimization across different layers and network nodes. Addressing these limitations, an alternative adaptive streaming architecture based on the TBS paradigm for scalable videos and designed to overcome the unique challenges found in a mobile environment is introduced in this chapter. Although the focus of the work will be toward OFDM-based radio technologies such as the LTE or the WiMAX where radio resources are divided into small time-frequency chunks, theoretically it can also be used with other radio technologies with shared radio resources such as the CDMA-based HSPA as well. A dedicated coordinated adaptation server is proposed to jointly adapt the video bitrates to the streaming users in the same cell periodically, taking into account their individual channel conditions, characteristics of the videos and the overall congestion situation. The adaptation is done such that the overall video quality is maximized given limited radio resources while at the same time being fair to other competing traffic and keeping the congestion at an acceptable level. This architecture will be referred to as the Coordinated Adaptive Streaming (CAS) from now on. Additionally, the proposed framework can also be simplified such that the adaptation is done independently for each user based on its own congestion situation alone. This slightly modified architecture is referred to as the Uncoordinated Adaptive Streaming (UAS). While the latter has the advantage of being a simpler architecture, e.g., requiring neither a dedicated adaptation server nor cross-layer information on the channel, its overall performance is slightly inferior than CAS as a trade off. Finally, complete analyses of

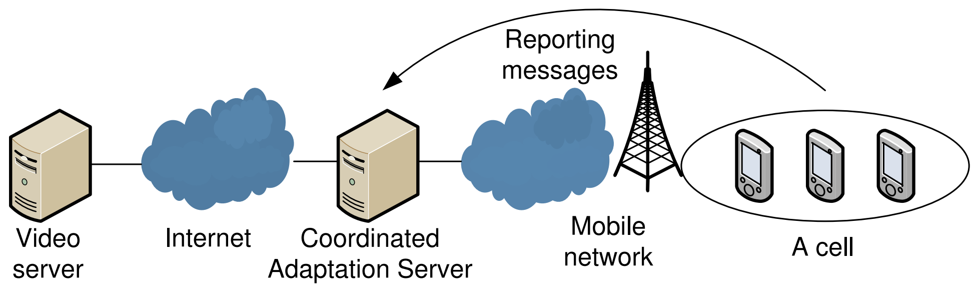


Figure 3.1: The architecture of the proposed CAS

the proposed framework in terms of, e.g., its complexity, the gain in video quality from performing coordinated adaptation and its relationships to other influential parameters are provided along with simulation results.

3.2 Overview of the Architecture

3.2.1 Coordinated Adaptive Streaming

The proposed CAS is an adaptive TBS architecture over RTP/UDP. To accommodate bitrate adaptation to the dynamic mobile environment, a coordinated adaptation server is proposed to be placed somewhere along the transmission path, e.g., at the edge of the mobile network, to periodically adapt the video streams. The adaptation interval, denoted by $t = 1, 2, \ldots,$ is set to be the duration of a GoP but theoretically it can be any integer multiple of the shortest adaptable period of the videos. The bitrates of the videos destined to users in the same mobile cell are adapted such that the overall video quality is maximized given the amount of available radio resources. To achieve such objective, the coordinated adaptation server needs up-to-date information on the mobile channels as well as the RD information of each interval of all the videos. Therefore, each streaming user needs to send periodic reports to the coordinated adaptation server. These reports, whose contents are to be discussed shortly, can be sent via UDP and treated like typical Application-layer packets. The reporting interval should be no longer than the adaptation interval to guarantee that the coordinated adaptation server always has the latest information before the next optimization round is due. Additionally, it is assumed that the RD information in terms of the video quality and bitrate at each operating point is made available to the coordinated adaptation server as well. Note that the metric for the video quality could be, e.g., the Quality of Experience (QoE), the used QP, etc. as a function of bitrate. In this work, the PSNR is used for simplicity. The overall architecture of the proposed CAS is shown in Figure 3.1.

As explained in Section 2.1, radio resources in an OFDM-based mobile network are divided into thousands of chunks in each second. The assignment of these chunks is controlled by a resource scheduler at the base station. The Adaptive Modulation and Coding (AMC) technique is then applied further to adjust the amount of Application-layer data and protection bits within each chunk, based on the estimated channel quality which is known to the scheduler. To avoid having cross-layer interfaces to the base station and being dependent on any specific technology, the external coordinated adaptation server does not have access to this channel quality information which is usually defined differently between various standards. However, for any streaming user $n = 1, 2, \ldots, N$, it is possible to obtain the amount of used radio chunks $\tilde{C_{n,t}}$ in each optimization round t at the mobile terminal itself via, e.g., a simple interface between the streaming client software and its Physical layer. Given that this interface is only between layers within the same mobile terminal, it should be theoretically easy for the hardware manufacturers to provide such feature to the operators and/or mobile application developers if there is enough incentive and demand for it. Nevertheless, a ratio between the latest average Application-layer throughput $\tilde{R}_{n,t}$ over $\tilde{C_{n,t}}$ in the same duration implies the quality of the channel and can be used as a generic channel quality indicator as follows.

$$\rho_{n,t} = \tilde{R}_{n,t}/\tilde{C_{n,t}} \tag{3.1}$$

For this purpose, both $\tilde{R}_{n,t}$ and $\tilde{C_{n,t}}$ are included in the reporting message from the user. For the HSPA technology, the concept remains valid but one has to compare $\tilde{R}_{n,t}$ with the number of timeslots assigned to the user instead of radio resource chunks. Note that the tilde sign over both $\tilde{R}_{n,t}$ and $\tilde{C_{n,t}}$ are meant to represent that these are actually "received" throughput and radio resources that the user has been given from the network respectively, and are not to be confused with other quantities to be introduced later on.

The coordinated adaptation server also measures the RTT delay in the identical manner to RTP [41]. Specifically, it keeps track of its local time $Ts_{n,k}$ when a RTP packet destined for user n with a sequence number k is forwarded to the mobile network. Upon generating a reporting message, the user includes in the message the sequence number K of the last received RTP packet along with the delay D_n measured from the reception time of that packet to the generation time of the report. Once the reporting message is received by the coordinated adaptation server at time Tr_n, the RTT delay for the user n at round t can be measured from

$$RTT_{n,t} = Tr_n - Ts_{n,K} - D_n \tag{3.2}$$

Additionally, define the normalized RTT deviation as a measure of an overall deviation from the target RTT limit, denoted as $RTT_{D,t}$ and RTT_{target} respectively.

$$RTT_{D,t} = \frac{\sum_{n=1}^{N} (RTT_{target} - RTT_{n,t})}{(N \cdot RTT_{target})} \tag{3.3}$$

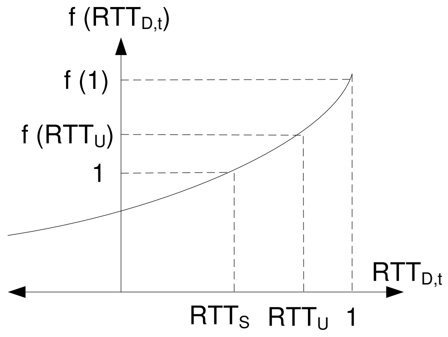


Figure 3.2: The resource scaling function for congestion control

The $RTT_{D,t}$ is a positive number no greater than one if the RTT delays for streaming users on average are less than the RTT_{target}. It becomes negative in the opposite case.

To perform congestion control, the total amount of allowed radio resources in the next period for all streaming users, denoted as $\hat{C}_{t+1}$, must be regulated. If the overall congestion for these users is light, then $\hat{C}_{t+1}$ should be increased to probe the network's capacity. On the contrary, $\hat{C}_{t+1}$ should be reduced if the cell becomes more congested. Let $\hat{C}_{t+1}$ be computed as follows.

$$\hat{C}_{t+1} = f\left(RTT_{D,t}\right) \cdot \sum_{n=1}^{N} \tilde{C_{n,t}} \tag{3.4}$$

Here, $f\left(RTT_{D,t}\right)$ is a scaling factor to adjust the $\hat{C}_{t+1}$ from the total amount of resources that have been used only by streaming users in the latest round t based on the previous $RTT_{D,t}$. The scaling function $f\left(RTT_{D,t}\right)$ should be designed such that streaming users have enough resources to continuously make a "jump" to their next operating points in the next round until they finally reach their highest layers when there is very low or no congestion. One of the possibilities, as shown in Figure 3.2, is to use the following exponential function.

$$f\left(RTT_{D,t}\right) = \omega^{\left(RTT_{D,t} - RTT_S\right)} \ ; \ \omega > 1,\ 0 \leq RTT_S < 1 \tag{3.5}$$

The RTT_S is a configurable constant which represents the value the $RTT_{D,t}$ converges to once the congestion in the cell is stable. The congestion control mechanism does not try to increase or decrease the $\hat{C}_{t+1}$ from the previously used amount of resources if the $RTT_{D,t}$ is equal to RTT_S, or in other words, $f\left(RTT_S\right) = 1$. If the overall congestion level increases such that $RTT_{D,t} < RTT_S$, the $f\left(RTT_{D,t}\right)$ becomes less than one and thus lowers the $\hat{C}_{t+1}$. On the contrary, the $f\left(RTT_{D,t}\right)$ becomes larger than one at an exponential rate if the congestion situation eases up. In case the congestion level is so low that $RTT_U \leq RTT_{D,t} \leq 1$, the $f\left(RTT_{D,t}\right)$ must be large enough to allow all users to upgrade to their next operating points. This is to prevent the system from being too conservative and to exploit the full network capacity. For this reason, the base of the exponent must be adjusted to the RD characteristics of the videos, specifically the different bitrate gaps between layers, at every adaptation interval. Let the average bitrate gap between operating points per streaming user at time t be $\Delta\bar{R}_t$ and the average channel quality be $\bar{\rho}_t$, from (3.1) it can be deduced that the average amount of resources needed per user to upgrade to the next layer is $\Delta\bar{C}_t = \Delta\bar{R}_t/\bar{\rho}_t$. The base of the exponent as a function of $\Delta\bar{C}_t$ and the amount of used resources can be derived at the beginning of each round as follows.

$$\left(f\left(RTT_U\right)-1\right)\cdot\sum_{n=1}^{N}\tilde{C_{n,t}} = N\cdot\Delta\bar{C}_t \tag{3.6}$$

$$\left(\omega^{(RTT_U-RTT_S)}-1\right)\cdot\sum_{n=1}^{N}\tilde{C_{n,t}} = N\cdot\Delta\bar{C}_t \tag{3.7}$$

$$\omega = \left[\frac{N\cdot\Delta\bar{C}_t}{\sum_{n=1}^{N}\tilde{C_{n,t}}}+1\right]^{1/(RTT_U-RTT_S)} \tag{3.8}$$

From experimental results, setting $RTT_S = 0.6$ and $RTT_U = 0.85$ provides a good balance between stability and responsiveness to the channels. Thus these constants are used throughout the rest of this thesis.

Once the $\hat{C}_{t+1}$ has been calculated for the next round, the coordinated adaptation server determines the best combination of bitrates $\vec{R}_{t+1} = (R_{1,t+1}, R_{2,t+1}, \ldots, R_{N,t+1})$, or equivalently the number of layers for streaming users at round $t+1$ by

$$\vec{R}_{t+1} = \arg\max\left(\sum_{n=1}^{N} Qlt_{n,t+1}\left(R_{n,t+1}\right)\right) \tag{3.9}$$

subject to

$$\sum_{n=1}^{N}\left(R_{n,t+1}/\rho_{n,t}\right) \leq \hat{C}_{t+1} \tag{3.10}$$

and

$$R_{min,n,t+1} \leq R_{n,t+1} \leq R_{max,n,t+1} \tag{3.11}$$

where the $R_{n,t+1}$ is the selected bitrate at the next interval for the user n whose minimum and maximum bitrates are represented by the $R_{min,n,t+1}$ and $R_{max,n,t+1}$ respectively. The

$Qlt_{n,t}\left(R_{n,t}\right)$ is the quality of the video, e.g., the PSNR, for the user n at interval t and bitrate $R_{n,t}$. The coordinated adaptation server then removes some layers from the video chunks until they have the desired bitrates before forwarding them to the mobile network.

3.2.2 Uncoordinated Adaptive Streaming

The UAS is different from the CAS in that the adaptation can be done at the originating server individually for each user, thus eliminating the need to have a centralized adaptation server. This is achieved by simply following the same adaptation procedure as for CAS, but with $N = 1$ for each streaming user instead. By considering only a single user n in (3.4) at a time, the congestion control equation becomes a simple relationship between the next allowed rate and its previous average throughput for each individual user by multiplying $\rho_{n,t}$ to it.

$$R_{n,t+1} = f\left(RTT_{D,n,t}\right) \cdot \tilde{R}_{n,t} \tag{3.12}$$

Note that the normalized RTT deviation and the scaling function, which are still the same as defined in (3.3) and (3.5) respectively but with $N = 1$, need an additional subscript n to distinguish between those of different users. In addition, since UAS does not need to know the amount of radio resources each streaming user used in the previous interval for the rate control in (3.12) anymore, the reporting message therefore only contains the average received throughput and information for measuring the RTT delay which are all measurable at the Application layer. The number of layers in the video for each user thus is selected to be as many as possible such that the corresponding bitrate is still within $R_{n,t+1}$.

3.3 Analysis on CAS

In this section, the CAS is formulated into an optimization problem of allocating limited system resources amongst streaming users to maximize the total video quality. The focus is especially on the case where the utility function is monotonically increasing and non-convex since most of the RD curves can be approximated as such [82]. The location of the optimum solution and its relationship with characteristics of the videos are studied. Other related works in optimization such as [83–85] however focus on developing algorithms for a broader range of monotonic utility function even without the concaveness requirement. One of such algorithms is the Polyblock algorithm [84,85] which converges to the optimum solution without the concaveness assumption. However, its rate of convergence has been shown to be slower and it is more complicated than other simpler algorithms that already work well for a concave utility function, e.g., the Iterative Efficient Set Approach (IEA) and the Steepest Ascend (SA). Thus, the attention is toward the latter two where a detailed analysis on their performances when applied to CAS is provided later on in this section. Note that since the analysis concerns a single adaptation period, the subscript t will be dropped from all variables unless stated otherwise.

In the following, the allocation of radio resources is of interest which is equivalent to bitrate allocation taking the individual channel quality into account according to (3.1). For convenience, the RD relationship of a video is transformed into a relationship between the video quality and the required amount of radio resource chunks instead, referred to as a Rsc-D curve. Thus, define $Q_n\left(C_n\right)$ as the resulting video quality for the user n given C_n radio resource chunks. The Rsc-D curve can be obtained from the RD curve by "scaling" the rate axis properly as follows.

$$Q_n\left(C_n\right) = Qlt_n\left(\rho_n \cdot C_n\right) \tag{3.13}$$

Let $Q_n\left(C_n\right)$ be a continuous and monotonically increasing function defined over $[C_{min,n}, C_{max,n}]$ and differentiable over $(C_{min,n}, C_{max,n})$ where $C_{min,n} = R_{min,n}/\rho_n$ and $C_{max,n} = R_{max,n}/\rho_n$. Additionally, let $Q_{tot}\left(\vec{C}\right) = \sum_{n=1}^{N} Q_n\left(C_n\right)$ and $\vec{C} = (C_1, C_2, \ldots, C_N)$. The resource constraints in (3.10) and (3.11) can be used to define a set $\widehat{U} \subset \Re_{+}^{N}$ of all feasible solutions as

$$\widehat{U} = \left\{ \vec{C} \;:\; \sum_{n=1}^{N} C_n \leq \hat{C},\ C_{min,n} \leq C_n \leq C_{max,n},\ n = 1, 2, .., N \right\}. \tag{3.14}$$

Taking a similar approach as in [83], it can be deduced that if there exist $\vec{C}_1, \vec{C}_2 \in \widehat{U}$ and $\vec{C}_1 \leq \vec{C}_2$, then $Q_{tot}\left(\vec{C}_1\right) \leq Q_{tot}\left(\vec{C}_2\right)$ due to the monotonicity of each individual Q_n and therefore Q_{tot} as well. Define the Pareto efficient set $\widehat{E} \subset \widehat{U}$ as

$$\widehat{E} = \left\{ \vec{C} \;:\; \vec{C} \in \widehat{U},\ \vec{C}' > \vec{C} \rightarrow \vec{C}' \notin \widehat{U} \right\}. \tag{3.15}$$

The efficient set $\widehat{E}$ basically contains all solutions that use the allowed resources $\hat{C}$ completely and thus can also be defined by the equality $\sum_{n=1}^{N} C_n = \hat{C}$ as well as the respective limits of each user's resource range. $\widehat{E}$ can be represented graphically as an upper-bounding "flat plane" of $\widehat{U}$ in an N-dimensional space as shown in Figure 3.3. In this example, $\widehat{E}$ is a line segment and a flat 2-dimensional plane when $N = 2$ and $N = 3$ respectively.

Subsequently, (3.9) can be written as an optimization problem over $\widehat{E}$ where $\vec{C}_{opt}$ denotes the optimizer as follows.

$$\vec{C}_{opt} = \underset{\vec{C} \in \widehat{E}}{\arg\max} \left(Q_{tot}\left(\vec{C}\right) \right) \tag{3.16}$$

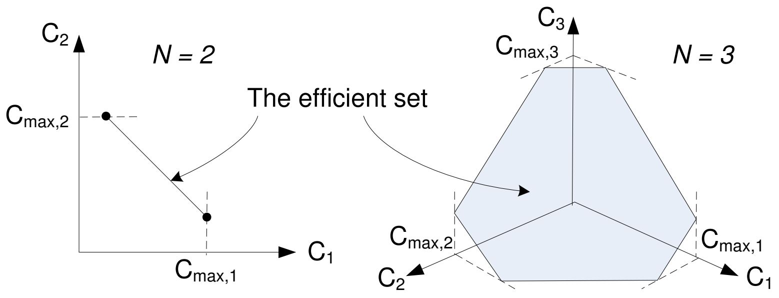


Figure 3.3: Examples of the efficient set $\widehat{E}$

The Lagrangian of (3.16) is therefore

$$J\left(\vec{C},\lambda\right) = Q_{tot}\left(\vec{C}\right) + \lambda \cdot \left(\sum_{n=1}^{N} C_n - \hat{C}\right). \tag{3.17}$$

Taking a partial derivative of (3.17) with respect to each C_n, the following set of equations can be obtained where $\vec{C}_{cr} = (C_1^{cr}, C_2^{cr}, \ldots, C_N^{cr})$ denotes a critical solution, e.g., a potential candidate for the $\vec{C}_{opt}$.

$$\left.\frac{dQ_n}{dC_n}\right|_{\vec{C}_{cr}} = -\lambda \ ; \ \forall n \tag{3.18}$$

The equations in (3.18) together with the resource constraint yield a set of $N+1$ independent equations to solve for $\vec{C}_{cr}$ and λ. They also imply that $\vec{C}_{cr}$ is a solution where the gradient of Q_{tot} is perpendicular to the plane described by $\widehat{E}$, e.g., it is the point where all users gain equal increase in their video quality per resource chunk.

To test whether $\vec{C}_{cr}$ is a local maximum, minimum or just a saddle point, the second-order derivative test using the Hessian matrix is used [86]. The Hessian matrix of Q_{tot}, denoted as $H\left(Q_{tot}\right)$, is a $N \times N$ matrix whose ij-entry is the second-order partial derivative $\partial^2 Q_{tot}/\partial C_i \partial C_j$. Since Q_{tot} is the sum of individual Q_n which is a function of only C_n, the Hessian matrix of Q_{tot} will always be a diagonal and symmetric matrix where entries outside the main diagonal line are all zero. Additionally, define a non-zero vector $\vec{z} = (z_1, z_2, \ldots, z_N)$ and the following matrix product.

$$\vec{z} \cdot H\left(Q_{tot}\right) \cdot \vec{z}^T = \sum_{n=1}^{N} z_n^2 \frac{\partial^2 Q_{tot}}{\partial C_n^2} \tag{3.19}$$

The Hessian matrix of Q_{tot} evaluated at $\vec{C}_{cr}$ is said to be positive definite if the product in (3.19) is greater than zero. In such case, $\vec{C}_{cr}$ is a local minimum. If the matrix is negative definite, e.g., the product is less than zero, $\vec{C}_{cr}$ is a local maximum. Finally $\vec{C}_{cr}$ is a saddle point if the matrix is indefinite and the product is zero.

In case $\vec{C}_{cr}$ cannot be found or $\vec{C}_{cr} \notin \widehat{E}$, e.g., there is no solution in $\widehat{E}$ which the gradient of Q_{tot} is perpendicular to $\widehat{E}$, one can always increase the Q_{tot} by moving from the current solution $\vec{C} \in \widehat{E}$ in the direction of the projection of the gradient on the plane until one of the "boundaries" of $\widehat{E}$ is reached. Reaching the boundary means one or more users have reached one of their C_{min} or C_{max} limits and can be considered fixed. Let K be the number of the users that have reached their limits, the remaining optimization problem is reduced to of $N - K$ dimension. This process which is essentially the concept of the IEA algorithm to be discussed later continues until the $\vec{C}_{cr}$ (with reduced dimension) is found or $N - K = 1$. For more rigorous proofs and details of the IEA algorithm see [84].

Although the existence of the optimum solution with concave and convex utility functions is generally known and not discussed in details in other related works, the following Sections 3.3.1 and 3.3.2 further elaborate in details the characteristics of the $\vec{C}_{opt}$ in the context of having different combinations of Rsc-D curves. This is to provide a complete picture of the problem and to serve as a basis for the later analyses.

3.3.1 Optimum Solution with Concave Rsc-D curves

Consider a special case where all Rsc-D curves are increasing and strictly concave functions, e.g., functions with $dQ_n/dC_n > 0$ and $d^2Q_n/dC_n^2 < 0$. To move from $\vec{C}_{cr} \in \widehat{E}$ to any solution $\vec{C}' \in \widehat{E}$, some users must use fewer resources so that others can be allowed to consume more. Let a user i be the one whose amount of allowed resources is downgraded and a user j be the one whose amount of allowed resources is upgraded, e.g., $C_i^{cr} > C_i'$ and $C_j^{cr} < C_j'$. Since the slopes of Q_i and Q_j evaluated at $\vec{C}_{cr}$ must both be equal to $-\lambda$, the new slopes evaluated at $\vec{C}'$ for both users can be written as the following.

$$\left.\frac{dQ_i}{dC_i}\right|_{\vec{C}'} = -\lambda + \int_{C_i^{cr}}^{C_i'} \frac{d^2Q_i}{dC_i^2} dC_i \tag{3.20}$$

$$\left.\frac{dQ_j}{dC_j}\right|_{\vec{C}'} = -\lambda + \int_{C_j^{cr}}^{C_j'} \frac{d^2Q_j}{dC_j^2} dC_j \tag{3.21}$$

The latter integration term in (3.20) integrates from C_i^{cr} to a smaller C_i' over a second-order derivative of Q_i which is less than zero. As a result, the new slope is larger than $-\lambda$. On the contrary, the latter integration term in (3.21) results in a real negative value, decreasing the slope of Q_j from $-\lambda$. Thus, it can be concluded that to move from $\vec{C}_{cr}$ to any other point $\vec{C}'$, the slopes of the Rsc-D curves evaluated at the new solution $\vec{C}'$ are

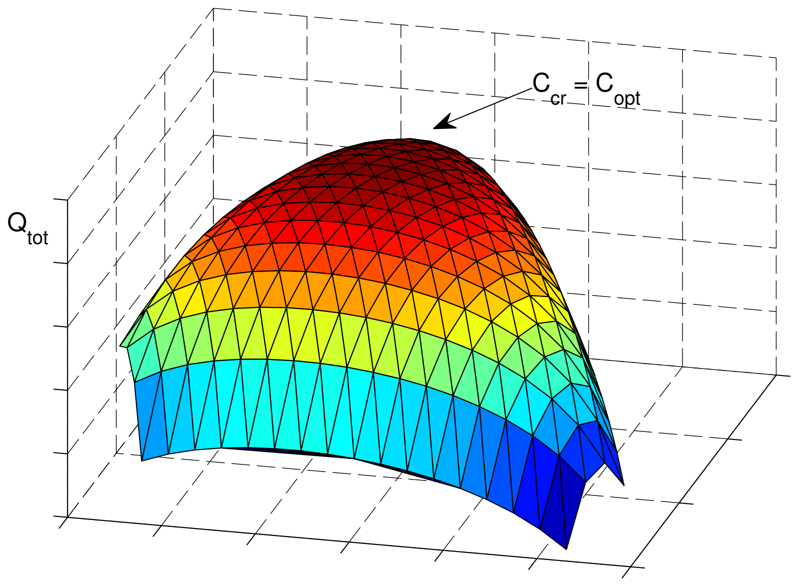


Figure 3.4: Value of Q_{tot} over $\widehat{E}$ for the concave case

not equal for all user pairs. Equation (3.18) is not fulfilled and thus $\vec{C}'$ cannot be another critical solution. Therefore, $\vec{C}_{cr}$ is a unique solution.

By testing $\vec{C}_{cr}$ with (3.19), it is found that the Hessian matrix is always negative definite and $\vec{C}_{cr}$ must be the global maximum as a consequence. If $\vec{C}_{cr} \in \widehat{E}$, then $\vec{C}_{opt} = \vec{C}_{cr}$. However, if $\vec{C}_{cr} \notin \widehat{E}$, the $\vec{C}_{opt}$ will be at the edge of $\widehat{E}$ closest to the $\vec{C}_{cr}$ as previously discussed. In either ways, there is no other local maximum in $\widehat{E}$.

Figure 3.4 shows an example for $N = 3$ of how the value of Q_{tot}, represented by the Z axis, varies over a 2D plane of $\widehat{E}$ which is now laid on the X-Y plane. All Rsc-D curves are modeled with strictly concave functions. This can be thought of as taking the plane shown in the right of Figure 3.3 where $N = 3$, laying it on the X-Y plane and plotting the corresponding values of Q_{tot} on top along the Z axis. In this example, $\vec{C}_{cr} \in \widehat{E}$, thus it is also the $\vec{C}_{opt}$ for CAS.

3.3.2 Optimum Solution with Non-Concave Rsc-D Curves

Assume all Rsc-D curves to be increasing linear functions, e.g., each curve is of the form $Q_n(C_n) = m_n \cdot C_n + C_{n0}$ where $m_n > 0$ and $C_{n0} \in \Re_+$. Under this assumption, equation (3.18) implies that the optimization problem can only be solved if $m_i = m_j = -\lambda$ for any pair of users i and j. In such an unlikely event, all solutions in $\widehat{E}$ are all critical solutions. However, evaluating the Hessian matrix of Q_{tot} at any of these points reveals that the

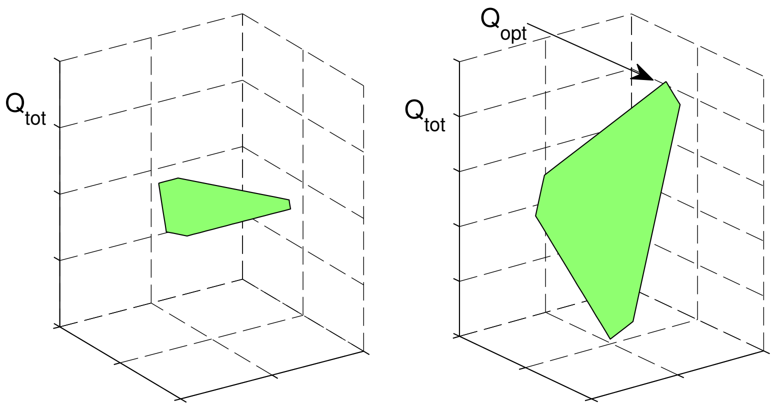


Figure 3.5: Value of Q_{tot} over the plane of the resource constraint for the linear case

matrix is always indefinite as the second-order derivative of any Q_n is zero, meaning they are all saddle points with equal Q_{tot}. The coordinated optimization server can pick any of these solutions to be the selected solution for CAS. This scenario is shown on the left of Figure 3.5.

On the contrary, (3.16) is not solvable and $\vec{C}_{cr}$ does not exist in a more likely scenario where the slopes for some or all of the Rsc-D curves are not equal. Thus, by moving in the direction of maximum gain in $\widehat{E}$, one can always increase the Q_{tot} until a boundary of $\widehat{E}$ is reached, in which the $\vec{C}_{opt}$ lies. Note that this scenario produces no other local maximum as depicted on the right of Figure 3.5.

In case of a mixture between concave and linear Rsc-D curves, the location of the $\vec{C}_{opt}$ for CAS depends on whether the slopes of the linear Rsc-D curves are the same or not. If the linear Rsc-D curves have the same slope, it is possible to determine the $\vec{C}_{cr}$ and the rest of the analysis is the similar to the case of all strictly concave curves, e.g., $\vec{C}_{cr}$ is the global maximum point and is therefore the CAS's $\vec{C}_{opt}$ if it lies within $\widehat{E}$. Otherwise $\vec{C}_{opt}$ is at the edge closest to the $\vec{C}_{cr}$. However, if the linear Rsc-D curves have different slopes, $\vec{C}_{cr}$ cannot be found and the CAS's $\vec{C}_{opt}$ is again at the edge. Note that for both cases, the $\vec{C}_{opt}$ is still the global maximum.

If there are both strictly convex and linear Rsc-D curves, the same analysis technique still applies but with some important differences. Firstly, the $\vec{C}_{cr}$, if can be found, will be a global minimum point instead and cannot be the $\vec{C}_{opt}$. Secondly, there could be more than one locally maximum solution at the edges of $\widehat{E}$, similar to the scenario with all convex Rsc-D curves, and the CAS's $\vec{C}_{opt}$ is therefore at one of these edges.

If some of the Rsc-D curves are convex, it is inconclusive whether $\vec{C}_{cr}$ is a maximum, minimum or simply a saddle point from (3.19). Additionally, using the same analysis method as in (3.20) and (3.21), one finds that if the user i to be allowed fewer resources has a convex curve and the user j to be given more resources has a concave curve, there can be several $\vec{C}_{cr}$ solutions where the slopes at those points are the same. Therefore, it is possible that there could be many locally optimum solutions in $\widehat{E}$ including some points along the edges but only one of them is the $\vec{C}_{opt}$.

3.3.3 Search Algorithms

In this section, two simple search algorithms commonly used in the literature are discussed in terms of their complexity, effectiveness and some design aspects that could effect their performances in a real deployment. Firstly, the theoretical concept is briefly summarized under the continuous Rsc-D curve assumption. Then the effects when the discrete nature of the Rsc-D curve is considered including potential failure cases where the algorithms fail to converge to the true global optimum are investigated.

Iterative Efficient Set Approach

The IEA has been studied in [83, 84] as a low-complexity search algorithm and found applications in, e.g., [48–50]. Its convergence to the global optimum is guaranteed with a monotonically increasing, continuous and concave utility function. The algorithm can be briefly summarized as follows. Let $\vec{C}^i$ be the selected solution at an iteration round i, $\alpha > 0$ and $\vec{C}^0 \in \widehat{E}$ is a random starting point. The algorithm moves the current selected solution to the next one by

$$\vec{C}^{i+1} = \vec{C}^i + \alpha P_{\vec{C}^i} \nabla Q_{tot}\left(\vec{C}^i\right). \tag{3.22}$$

The $P_{\vec{C}^i}$ denotes the orthonormal projector onto the tangent space of $\widehat{E}$ at $\vec{C}^i$. Verbally, $\vec{C}^{i+1}$ is shifted from $\vec{C}^i$ in the direction of the projection of the ∇Q_{tot} onto the tangent space with α controlling the rate of change. Since $\widehat{E}$ is a flat plane in the N-dimensional space, a projection onto a tangent space at any $\vec{C}^i \in \widehat{E}$ is also equivalent to a projection onto $\widehat{E}$ itself. This updating process continues until $\vec{C}^{i+1} - \vec{C}^i$ is less than a desired threshold before the iteration terminates.

If the continuous assumption is replaced by the discrete nature of the Rsc-D curves, the plane $\widehat{E}$ is also replaced by a set containing all largest discrete solutions that lie "just under" or on the plane $\widehat{E}$. Denote this discrete efficient set as $\widehat{E}_d$. The same concept of climbing up the steepest path on $\widehat{E}_d$ from a random starting point $\vec{C}^0 \in \widehat{E}_d$ still applies. In each iteration, the algorithm downgrades the user with the smallest loss in the video quality and gives the released resources to the one with the largest quality gain and continues until additional quality increase is not possible.

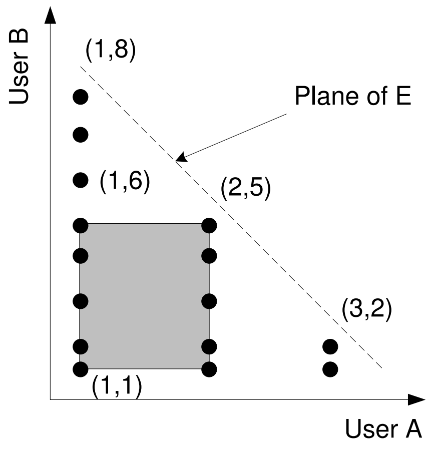


Figure 3.6: An example difficult case for search algorithms

The worst-case complexity estimation of the IEA, denoted as $O\left(IEA\right)$, is obtained from when it has to search through all solutions in $\widehat{E}_d$. Obviously, the cardinality of $\widehat{E}_d$, denoted as $\left|\widehat{E}_d\right|$, depends on the number of users N, the number of layers per user L and the amount of allowed resources $\hat{C}$. Although an analytical relationship between $\left|\widehat{E}_d\right|, N, L$ and $\hat{C}$ is difficult to obtain, the worst-case complexity can still be loosely bounded as follows. Assume each layer requires the same amount of resources for any user for simplicity. When $\hat{C}$ is only enough to support just base layers for all users, $\left|\widehat{E}_d\right| = 1$. As $\hat{C}$ grows to a certain value that is enough to support one user at its highest layer and the rest at their base layers, it is possible to show that $\left|\widehat{E}_d\right| = C_{N-1}^{L+N-2}$. Beyond this point, this formula does not hold as the users are limited to have at most L layers. $\left|\widehat{E}_d\right|$ still continues to grow but at a slower rate, then becomes saturated and shrinks back to one when $\hat{C}$ is large enough for everyone to have their highest layers. Thus a rather loose bound on the worst-case complexity is $O\left(IEA\right) \approx O\left(\left|\widehat{E}_d\right|\right) \leq O\left(C_{N-1}^{L+N-2}\right)$

The discrete IEA works well and guarantees the convergence to the global optimum with a concave utility function as long as the gaps between layers and different users are comparable. Convergence to one of the local optimums is also guaranteed if the concaveness assumption is dropped. However, if the channels are at the extremes and/or the bitrate gaps between layers are vastly different amongst users, the performance of the algorithm can be severely affected. Consider a representative failure scenario in Figure 3.6 where $N = 2$ with three and eight operating points for user A and B respectively. Let (a, b) represent a solution where a and b layers are selected for the user A and B. If the random starting point in $\widehat{E}_d$ is the point $(1, 8)$ and $\vec{C}_{opt} = (2, 5)$, the algorithm will not be able to downgrade the user B by one layer and use the released resources to upgrade the user A to reach $\vec{C}_{opt}$. Thus, it is forced to exit and incorrectly conclude the starting point $(1, 8)$ is the best solution. A quick fix to this shortcoming would be to downgrade a user as much as necessary so that at least the M^{th} least demanding user amongst the other $N-1$ users in terms of needed resource to upgrade itself by a layer can be upgraded. Note that $M < N$. Although this prevents the algorithm from terminating prematurely, its complexity surely increases as a result.

By starting the algorithm at $(3,2)$, the discrete IEA can potentially reach the $\vec{C}_{opt}$ since downgrading the user A releases enough resources to upgrade the user B by up to three layers. However, here lies another potential problem as well as one needs to decide which other users the freed resources should be allocated to in case $N > 2$ and the additional resources are enough to upgrade multiple users by several layers. A possible solution would be to use the Steepest Ascend algorithm (to be discussed next) to distribute the released resources from the downgraded user with the expense of increased complexity.

Steepest Ascend

The algorithm discussed in this section is a slight variation from the original IEA in [83,84]. Typically, the IEA starts from a random point on $\widehat{E}$ and works its way toward $\vec{C}_{opt}$ while being on $\widehat{E}$ all the time. Alternately, one can make a slight variation from the IEA by starting at the lowest solution $\vec{C}^0 = (C_{min,1}, C_{min,2}, \ldots C_{min,N})$ and "ascending" toward $\widehat{E}$ by following the path with steepest gradient. To distinguish between this slight variation from the IEA, this algorithm is referred to as the Steepest Ascend (SA) algorithm from now. The SA also guarantees a convergence to the global optimum with a monotonically increasing, continuous and concave utility function. The convergence is also guaranteed for one of the local optimums if the Q_{tot} is not concave. Let $C^i_{used} = \sum_{n=1}^{N} C^i_n$ be the total consumed resources by the current selected solution $\vec{C^i}$. If $\hat{C} - C^i_{used} > 0$, the SA improves the selected solution to be closer to the plane $\widehat{E}$ by

$$\vec{C}^{i+1} = \vec{C^i} + \alpha \nabla Q_{tot}\left(\vec{C^i}\right). \tag{3.23}$$

This process continues until the allowed resources $\hat{C}$ are used up or no further upgrade is possible, e.g., all users are at their highest layers. The upgrading process basically follows the gradient of the Q_{tot} from the current selected solution toward the plane $\widehat{E}$ where the rate of the ascend is controlled by α.

In the discrete scenario, the SA assigns more resources to upgrade the user with the steepest Rsc-D curve at the current selected solution in each round. If the one with the steepest curve cannot be upgraded, either because the remaining resources are not enough or the highest layer has been reached, the next steepest user is considered for upgrading instead and so on. The iteration continues until no more improvement is possible and the plane $\widehat{E}_d$ has been reached.

N	L	Exhaustive $\left(L^N\right)$	IEA $\left(\left\|\widehat{E}_d\right\|\right)$	SA $(N \cdot L)$
3	5	125	15	15
6	5	15625	126	30
10	5	9765625	715	50

Table 3.1: Complexity comparison between algorithms

The worst-case complexity for the discrete SA algorithm is when it has to ascend from $\vec{C}^0$ all the way to $\vec{C}_{opt} = (C_{max,1}, C_{max,2}, \ldots C_{max,N})$. This requires the largest number of iterations as the algorithm has to iteratively increase the number of layers for each user by one at a time to the top layer. Therefore we have $O(SA) \approx N \cdot L$. Note that this worst-case bound on $O(SA)$ which grows linearly with N and L is still significantly lower than $O(IEA)$ which grows with factorials of N and L as discussed in the previous section.

The discrete SA algorithm also has limitations similar to the IEA algorithm. Consider the same situation in Figure 3.6 where the bitrate gaps are significantly different between users. To make the ascend from the starting solution $(1,1)$ to the $\vec{C}_{opt} = (2,5)$, the algorithm has to upgrade $\vec{C}^i$ such that $\vec{C}^i \leq (2,5)$ for all iteration rounds. More generally, $\vec{C}^i$ has to stay within the polyblock of $(2,5)$, represented here with the shaded area. If, for example, $\vec{C}^i$ is selected to be $(1,6)$ at one of the iteration rounds due to the greater efficiency toward this solution from $(1,5)$ than the efficiency toward the true optimum at $(2,5)$, then the ascend will be unrecoverable and the global optimum is missed. Nevertheless, this failure case rarely happens. The performance of the discrete SA is in fact far superior to the IEA and comparable to the exhaustive search as will be shown by simulations later on. Additionally, its complexity is much less than the IEA's as demonstrated in Table 3.1 for N users and L layers per user.

3.3.4 CAS and the Base Station's Resource Scheduler

This section discusses the CAS's performance and its dependency on the type of the base station's resource scheduler. Specifically, it is of interest as to what influences the scheduler's sensitivity to the queuing delay at the base station has to the CAS's ability to adapt the streaming users' bitrates.

To understand how the scheduler's ability to adjust its resource assignments to the delay is important to CAS, one must understand how an external adaptation entity influences the scheduler's bitrate allocations first. For an architecture with no congestion control mechanism to regulate the amount of video streaming resources as in [48–51], the adaptation entity must know exactly how many resources are available in the system to all the users, including non-streaming ones. It can therefore control the bitrates to all users such that the available resources are completely used up within one adaptation period, having no accumulation of delay and all buffers are empty at the end of each round. From the scheduler's point of view, the queues for users that are less-preferred by the adaptation entity are shorter and tend to be empty faster. Once these queues are empty, the scheduler is forced to allocate remaining resources to other non-empty queues which belong to other more-preferred users, destined to have higher bitrates by the adaptation entity. Therefore, the bitrate allocations can be externally manipulated via controlling how much data there are in different queues, regardless of the actual scheduler's decision metric. This architecture can also theoretically work with non-QoS schedulers such as the Round Robin (RR) or the Maximum Throughput (MT) variants. The drawback with this method, amongst other things, is that the adaptation entity needs to oversee the bitrates for every user in the cell to get a complete and accurate picture of the resource consumption in each round. This requires extensive cross-layer information exchange between the two entities and complicates the optimization problem as discussed earlier.

On the contrary, an architecture with a congestion control mechanism such as CAS does not have a detailed picture of how much bitrate each user is requesting in the cell, especially the non-streaming ones. The coordinated adaptation server instead makes a guess how many resources are available to the streaming traffic class based on the measured RTT delays in order to probe the channels. It is therefore likely that not all queues can be emptied at the end of each round and there are accumulations of delay for some or all streaming users. As a result, manipulation of the scheduler's resource allocations externally cannot be done via controlling the sequence of queues to go empty, but instead via controlling the queue length since users that the coordinated adaptation server wants to have higher bitrates tend to have more packets waiting in their queues than the less-preferred ones, thus suffering larger queuing delay as a result. If the scheduler does not take the packet delay into its metric calculation, e.g., the RR and the MT schedulers or the PF scheduler with $F_{delay} = 1$ as discussed in Section 2.1.2, the CAS's performance as well as the coordination gain are expected to be reduced.

3.4 Analysis on UAS

As shown in (3.12), the rate selection for a UAS user is independent of the characteristics of its video, but is influenced only by the averaged received throughput and delay. The general solution for the UAS users, denoted as $\vec{C}_{UAS}$, is therefore derived from the understanding on how the network, or specifically the base station's resource scheduler, allocates resources amongst them based on their channel conditions and QoS requirements.

To derive $\vec{C}_{UAS}$, assume that the used resource scheduler is the PF scheduler discussed earlier in Section 2.1.2. The F_{QoS} and F_{delay} correction factors in (2.1) are implementation specific, but are usually a function of the relative GBR to the current throughput and a function of the delay respectively. However, both correction factors are at the moment set to be the same as those used in the simulation testbed (to be discussed shortly in the next section) for the purpose of demonstration. Specifically, the F_{delay} is set to be an exponential function that grows larger than one once the delay exceeds a preset guaranteed delay whereas the F_{QoS} is a function of the GBR for the user n and the average throughput up to the allocation time of radio resource chunk k, denoted as γ_n and $\tilde{r}_{n,k}$ respectively, as follows.

$$F_{QoS} = \left(\frac{\gamma_n}{\tilde{r}_{n,k}}\right)^5 \tag{3.24}$$

Generally, the users with relatively higher metrics than the average are likely to be given more resources, causing their metrics to decrease and vice versa. Although the instantaneous metrics of different users are unlikely to be the same at any particular radio resource chunk, their average values over a relatively longer period, e.g., over the adaptation interval of the UAS, tend to be kept equal amongst all users. Thus, for any pair of users i and j, the following relationship can be established.

$$mean\left(\tilde{m}_{i,k}\right) = mean\left(\tilde{m}_{j,k}\right) \tag{3.25}$$

$$\frac{\rho_i \gamma_i^5}{\tilde{R}_i^6} \cdot F_{delay}\left(RTT_i\right) = \frac{\rho_j \gamma_j^5}{\tilde{R}_j^6} \cdot F_{delay}\left(RTT_j\right) \tag{3.26}$$

At this point, the instantaneous quantities specific for the radio resource chunk k have been averaged and transformed into long term quantities instead. Thus the chunk index k has been dropped as a result. In addition, if it is assumed that most of the RTT delay is the downlink queuing delay at the base station and the uplink delay is negligible, then F_{delay} can be estimated as a function of the RTT delay as well.

In the case where Rsc-D curves are continuous and defined over $\Re$, each UAS user always has enough data to send up to any allowed bitrate. If the changes in the channel situation are relatively gradual compared to the adaptation interval of UAS, it can be assumed that the allowed bitrate from the UAS congestion control converges to the allocated bitrate from the PF scheduler, e.g., $R_n \approx \tilde{R}_n$. This is because both the UAS congestion control and the PF scheduler are self-adjusting toward each other, e.g., the UAS server tries to reduce the R_n if the delay situation is getting worse while the PF scheduler also tries to increase the $\tilde{R}_n$ to compensate and vice versa in the opposite situation. Thus, the following linear relationship between the amount of resources for any user pair i and j can be derived from (3.1) and (3.26).

$$\left(\frac{\gamma_i^5 F_{delay}\left(RTT_i\right)}{\rho_i^5}\right)^{1/6} C_j = \left(\frac{\gamma_j^5 F_{delay}\left(RTT_j\right)}{\rho_j^5}\right)^{1/6} C_i \tag{3.27}$$

Additionally, define C_{tot} as the total amount of radio resource chunks available for all UAS users in the cell. This could be the actual total number of chunks there are in the system's bandwidth or a fraction of it as determined to be appropriate and fair to other non-streaming traffic classes by the scheduler. The following additional resource constraint can then be established.

$$\sum_{n=1}^{N} C_n \leq C_{tot} \tag{3.28}$$

Equations (3.27) and (3.28) give a total of N linearly independent equations to solve for all N users. Let $\Lambda_i = \left(\frac{\gamma_i^5 F_{delay}(RTT_i)}{\rho_i^5}\right)^{1/6}$, these equations can be arranged in a form of a matrix equation which can be solved numerically as follows.

$$\begin{bmatrix} \Lambda_2 & -\Lambda_1 & 0 & \cdots & 0 \\ 0 & \Lambda_3 & -\Lambda_2 & & \vdots \\ \vdots & & & \ddots & 0 \\ 0 & \cdots & 0 & \Lambda_N & -\Lambda_{N-1} \\ 1 & 1 & 1 & \cdots & 1 \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \\ \vdots \\ C_{N-1} \\ C_N \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ C_{tot} \end{bmatrix} \tag{3.29}$$

Note that these equations and the resulting $\vec{C}_{UAS}$ are independent from the characteristics of the videos, but are dependent solely on the channel qualities, the QoS settings and the RTT delays. It is therefore unlikely that the CAS's $\vec{C}_{opt}$ would be the same as the $\vec{C}_{UAS}$, implying that the coordination gain exists and is greater than zero.

If the continuous assumption is dropped and the Rsc-D curve is defined over the range $[C_{min,n}, C_{max,n}]$, the $\vec{C}_{UAS}$ can still be solved using (3.27) and (3.28) as before but with some modifications. First, it is possible that some users might have already reached their highest layers using just a fraction of all resources originally given to them by the scheduler. The left-over resources must then be redistributed to others that still have more data to send. This requires an iterative calculation amongst the remaining "active" users to further refine the final $\vec{C}_{UAS}$. The iteration is repeated until all users are at their highest layers or the remaining resources are not enough to upgrade anyone. Note that since there are discrete operating points on the Rsc-D curves, it is likely that $\sum_{n=1}^{N} C_n < C_{tot}$ and there are left-over resources in each round.

3.5 Simulation Testbed and Evaluation Metrics

3.5.1 Simulation Testbed

The simulation testbed is composed of three main components - the Video Servers, the LTE simulator and the Users modules as depicted in Figure 3.7. The Video Servers module is a

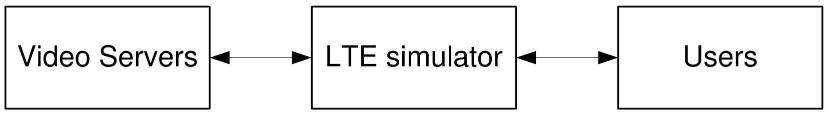


Figure 3.7: A diagram of the adaptive TBS testbed's components

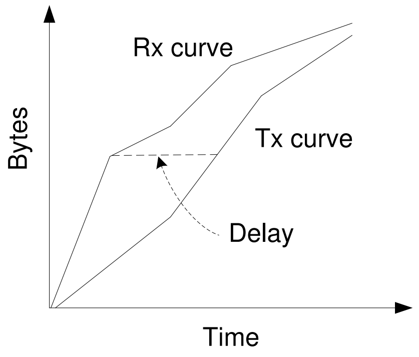


Figure 3.8: The base station's traffic curves for the accumulated amount of Bytes received from the server (Rx) and transmitted to the user (Tx)

video traffic generator representing external video servers and the coordinated adaptation server that feeds the adapted video streams to the mobile network. The LTE simulator simulates the behaviors of an LTE mobile cell. The mobile streaming users within the same cell are represented by the Users module. These users generate reports to the coordinated adaptation server in case of the adaptive streaming simulation as previously discussed in Section 3.2.

Two types of the LTE simulator were used to conduct simulations in this work. The first one is a real-time LTE simulator for application testing [22]. This real-time simulator allows realistic simulations of the behaviors of the LTE protocols, the radio channels, resource contention with other internally-generated cross-traffic users and the resulting congestion and losses in the cell. This simulator receives video data packets transmitted from the Video Servers module and either drops or forwards them to the video streaming users with appropriate delay. These streaming users, residing in the Users module, reconstruct the received and possibly corrupted video files for later decoding and quality evaluation.

Another type of the LTE simulator used was a simplified mathematical model of an LTE cell which only simulates its bitrate assignments to users in each adaptation period. It works iteratively to refine the resource allocation by the PF scheduler and updates the corresponding RTT delays and queue sizes for all users. A general concept on how such a simple simulator can be constructed is discussed only briefly as follows as it is not the main focus of this work. However, it is detailed enough to serve as a guideline for anyone who might be interested.

The mathematical model of the LTE can be thought of as a set of equations that reacts to inputs, in this case the incoming video bitrates from the servers and channel quality indicators of the users' channels, and computes the output network responses which are the resulting bitrate allocations from the PF scheduler and the accumulation of the delay in each adaptation round. First, the Rx and Tx traffic curves, representing the accumulated amount of data the base station has received from the coordinated adaptation server and the accumulated amount of data forwarded to a mobile user respectively in Bytes, are constructed for each user in the cell and updated at the beginning of each round. An example of these traffic curves from the base station's perspective are given in Figure 3.8 where the base station receives the first Byte at time $t = 0$. The downlink queuing delay for each user can be obtained simply from the horizontal distance between the two curves. Similarly, the queue size for each user is obtained from the vertical distance between the two curves. With other information such as the QoS settings and the channel qualities available, the allocated bitrates for all users by the PF scheduler can be computed using (3.26). This process is done iteratively so that any remaining resources from some users that experience buffer underrun are redistributed again to other users with more data to send, similar to the processes used to compute $\vec{C}_{UAS}$. Once the bitrate allocations have been determined, the traffic curves and the corresponding delays are updated. Note that for a streaming application where most of the traffic load is in the downlink direction while only control messages and voice traffic usually occupy the uplink, the RTT delay can be a approximated from the downlink queuing delay as the uplink delay is relatively negligible. The streaming users are subsequently notified of their received throughputs and the RTT delays in this adaptation round so that they can generate proper reports to the coordinated adaptation server before continuing with the next round.

The LTE model only needs to know the incoming video bitrates from the coordinated adaptation server to create imaginary traffic curves for its internal calculation without requiring the actual packets to traverse the LTE simulator module to the Users module. The servers do not necessarily send actual video packets to the LTE model nor do the streaming users actually receive video packets for decoding. This allows more flexibility in that any virtual video with any desired RD characteristics in terms of, e.g., its shape, the number of operating points and the bitrate range, can be simulated with ease. This is very difficult with actual video streaming using the real-time LTE simulator due to the difficulties of finding and encoding real video materials to have the desired RD characteristics. Additionally, the LTE model is able to run simulations faster than real time. Long simulations equivalent to several hours of video streaming can be done in a shorter period of time. This testbed configuration therefore allows the results to be more statistically diverse and avoids decoding and handling of large video files afterward as well. However, the degradation effects to the video quality due to packet losses could not be evaluated since the current LTE model does not simulate such loss event.

3.5.2 Normalized Total Quality

In order to compare the performance between the CAS and the UAS in terms of the overall video quality, an evaluation metric that reflects the result from adaptation and scalability of the videos, but is not influenced by the fact that different videos can have different absolute qualities and adaptation ranges is needed. In other words, if there are two simulations with identical settings except that one has videos with the adaptable PSNR range from 30 to 35 dB whereas another one is from 25 to 40 dB, the values of this metric from both simulations should be similar. It should reflect only the relative overall quality improvement over the base layer and should not be affected by the different baseline qualities. Thus, define a normalized total video quality as

$$Q_{norm} = \frac{\sum_{n=1}^{N}\left(\bar{Qlt}_n - \bar{Qlt}_{min,n}\right)}{\sum_{n=1}^{N}\left(\bar{Qlt}_{max,n} - \bar{Qlt}_{min,n}\right)}. \tag{3.30}$$

The $\bar{Qlt}_{min,n} = mean\left(Qlt_{n,t}\left(R_{min,n,t}\right)\right)$ is the time-averaged video quality at the base layer of the user n over the entire length of the video. The $\bar{Qlt}_{max,n} = mean\left(Qlt_{n,t}\left(R_{max,n,t}\right)\right)$ is the time-averaged video quality at the highest layer and finally $\bar{Qlt}_n = mean\left(Qlt_{n,t}\left(R_{n,t}\right)\right)$ is the time-averaged received video quality. This metric depends only on the ratio between the overall quality improvement from the baseline quality over the total adaptation range, and is therefore suitable for comparing the quality between different simulation runs.

3.5.3 Normalized Total Channel

Similar to the normalized total quality, a metric that represents the overall channel situation of all users is needed. It should represent the instantaneous relative capacity of the cell to support the requested videos taking into account the video bitrates and the bandwidth of the system. Define a normalized total channel for any adaptation round t as follows

$$P_{norm,t} = \frac{\sum_{n=1}^{N}\rho_{n,t}}{N\cdot\rho_{best}}. \tag{3.31}$$

The ρ_{best} which represents the required channel quality to support the bitrate of the highest enhancement layers using all the radio resources available to the streaming traffic is defined as

$$\rho_{best} = \frac{\sum_{n=1}^{N}\left(mean\left(R_{max,n,t}\right)\right)}{C_{tot}}. \tag{3.32}$$

Consequently, the $P_{norm,t}$ equal to or greater than one means the cell is on average capable of supporting all streaming users with their highest video layers. On the contrary, video bitrates for some users must be adapted and reduced when the $P_{norm,t}$ is less than one.

3.6 Simulation Results and Analyses

This section discusses simulation results and analyses on various factors that affect the performance of CAS and UAS. Most of the simulations were done with the following common settings, unless stated otherwise. The mathematical model of the LTE as discussed in Section 3.5.1 is used in the testbed for most of the simulations to allow flexibility in simulating imaginary videos with desired RD characteristics. There are two to four streaming users in the cell where the C_{tot} is the total resources in the system. The channel $\rho_{n,t}$ for a user n is randomized from 40% to 130% of the ρ_{best} in each adaptation period. The simulation length is $10,000$ periods while a period is set to be 32/30 seconds, representing the duration of two GoP's, each having 16 frames and being played at 30 fps. This results in approximately three hours worth of video streaming time. Finally, all the results in the subsequent analyses have been averaged over the entire simulation length, hence the time index t is dropped for simplicity.

3.6.1 Operating Regions

The operating regions refer to different ranges of the normalized total channel P_{norm} as defined in (3.31) where the adaptation actions, or the lack thereof, take place differently. Consider Figure 3.9 from simulations with three users each having five operating points and $C_{tot} = 8,000$ chunks. The top figure shows three resource-related plots versus the normalized total channel. These plots are the amount of allowed resources $\hat{C}$ from the CAS's congestion control mechanism, the amount of needed resources to support the chosen $\vec{C}_{opt}$, and the actual amount of resources given to CAS users by the PF scheduler. These lines are labeled in the figure as "CAS allowed", "CAS needed" and "CAS used" respectively. The middle figure shows the same plots but from the UAS simulations with otherwise the same settings. The bottom figure shows the value of the $f\left(RTT_{D,t}\right)$ from the CAS's congestion control and the average value of the $f\left(RTT_{D,n,t}\right)$ over all N UAS users versus the normalized total channel.

The easiest way to interpret the meaning of Figure 3.9 is to start from the best channel situation, e.g., when $P_{norm} > 1$. In this region, referred to as the "Max quality" region, the channels are in a very good situation. All users are able to receive their highest layers without having to use all the available C_{tot} for streaming, according to the definition of the P_{norm}. The amount of needed resources as well as the amount of resources given by the scheduler are therefore equal to each other and less than the C_{tot}, e.g., the better the channel, the fewer radio chunks are needed to carry the same amount of data. The RTT

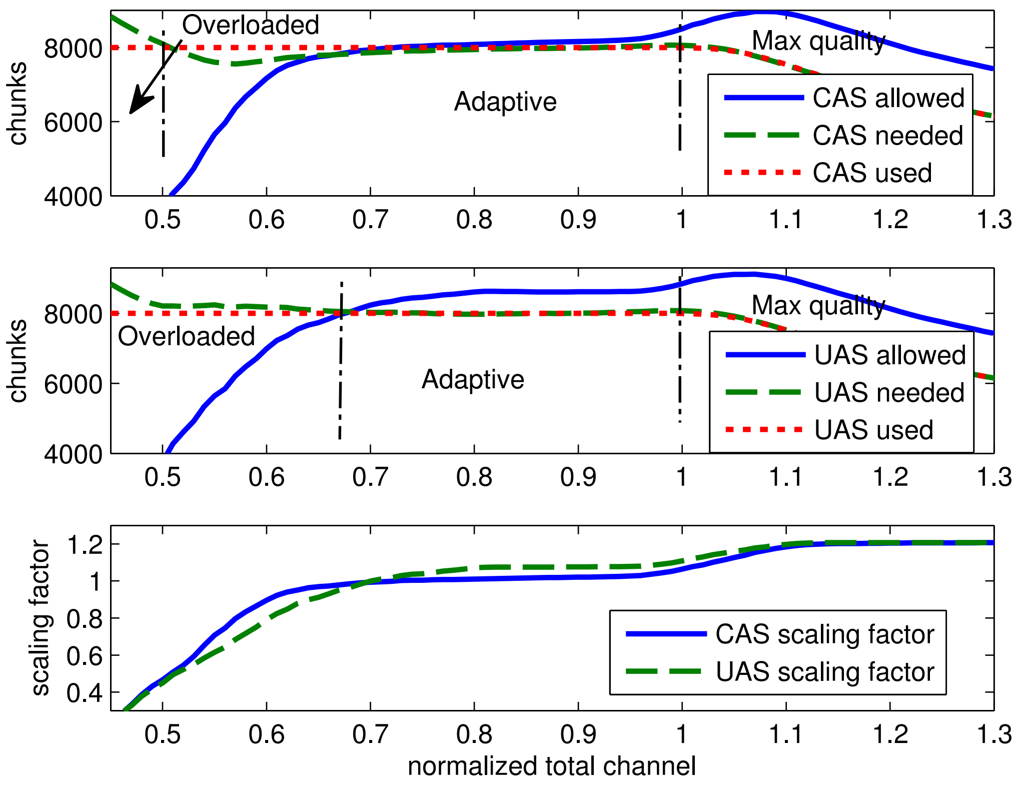


Figure 3.9: Operating regions for the CAS and the UAS

delay for each user is also close to zero, prompting both the CAS's and UAS's congestion control mechanisms to try increasing the amount of allowed resources in an attempt to probe for the network's capacity. This can be seen from the values of the scaling functions in the bottom figure which are greater than one. However, since all users are already at their highest layers, the amount of required resources cannot be increased to match with the allowed resources and thus is why there are discrepancies between these lines. Additionally, since both the CAS's and UAS's solutions are the same, no coordination gain is expected in this region.

If the P_{norm} deteriorates below one, some or all users cannot support their highest layers anymore and the adaptation takes place. This region is referred to as the "Adaptive" region. For CAS, the RTT delay begins to build up to the point where the congestion control does not try to increase nor decrease the amount of allowed resources anymore, but maintains it to be more or less equal to the C_{tot}. Thus, the $f\left(RTT_{D,t}\right)$ is approximately close to one. A similar explanation also applies to the UAS case, albeit the total amount of allowed resources from each individual user's congestion control mechanism is slightly larger than the amount of needed and given resources. This is due to the discrete nature of the RD curves used in the simulation and the way the UAS does the congestion control separately for users. Specifically, it is impossible for a user to find the operating point that completely uses all the resources allowed by its congestion control mechanism. There are always unused resources left from each user which result in a slightly lighter traffic load than allowed in the cell. As a consequence, the RTT delay for each user is also slightly lower and each of them tries to increase the amount of their allowed resources to compensate, although most likely they will end up not being able to use them all again and the system becomes stable at this state. This effect from having discrete RD curves also applies to CAS as well, although to a lesser extend since CAS users are adapted coordinately by the centralized adaptation server and share the same $\hat{C}$ resource pool. The unused resources from all users are accumulated together and are likely to be large enough to upgrade some users by a few additional layers, instead of wasting away unused as in the UAS case. In other words, one can say the resources are used more efficiently in this region by CAS than UAS.

Once P_{norm} continues to degrade further to the point where the amount of allowed resources is not enough to support even the base layers for some or all users, the amount of needed resources will be larger than the allowed resources. This is because the servers are forced to ignore the congestion control mechanism and send at least the base layers to the users. If the amount of needed resources to support their base layers is greater than what the scheduler can give them, represented here by the "CAS used" and "UAS used" lines in the plots, then the system will become unsustainable. The region below this point downward is referred to as the "Overloaded" region where the RTT delay continues to grow uncontrollably and will finally cause buffer overflow and losses at the base station if the situation persists. Simulation results show that the transition into the overloaded situation happens at a lower P_{norm} for the CAS case, implying that the system is more tolerable to channel degradation than UAS. This is because the allowed resources are shared among all the CAS users. Those with exceptionally bad channels can be temporarily allocated more resources by reducing the number of layers of other good users. Obviously this is not possible for UAS users that adapt their bitrates individually. Note that since the testbed with the mathematical model of the LTE does not simulate losses and the effects they have on the video quality, Q_{norm} in this region from this testbed configuration does not accurately represent the system and is omitted in the later simulations.

3.6.2 Different RD Characteristics

This section discusses the coordination gain from simulations with various types of the RD curves for three streaming users. Figure 3.10 on the left shows three types of simulated non-convex RD curves used in simulations, specifically a linear RD curve, a quadratic RD curve and a RD curve constructed using the mathematical model described in [82]. For each type of the mentioned RD curve, a set of two simulation runs was conducted - one simulation for CAS and another for UAS. The RD curves for all three users in each simulation set were of the same type, had the same bitrate range and five operating points. They were, however,

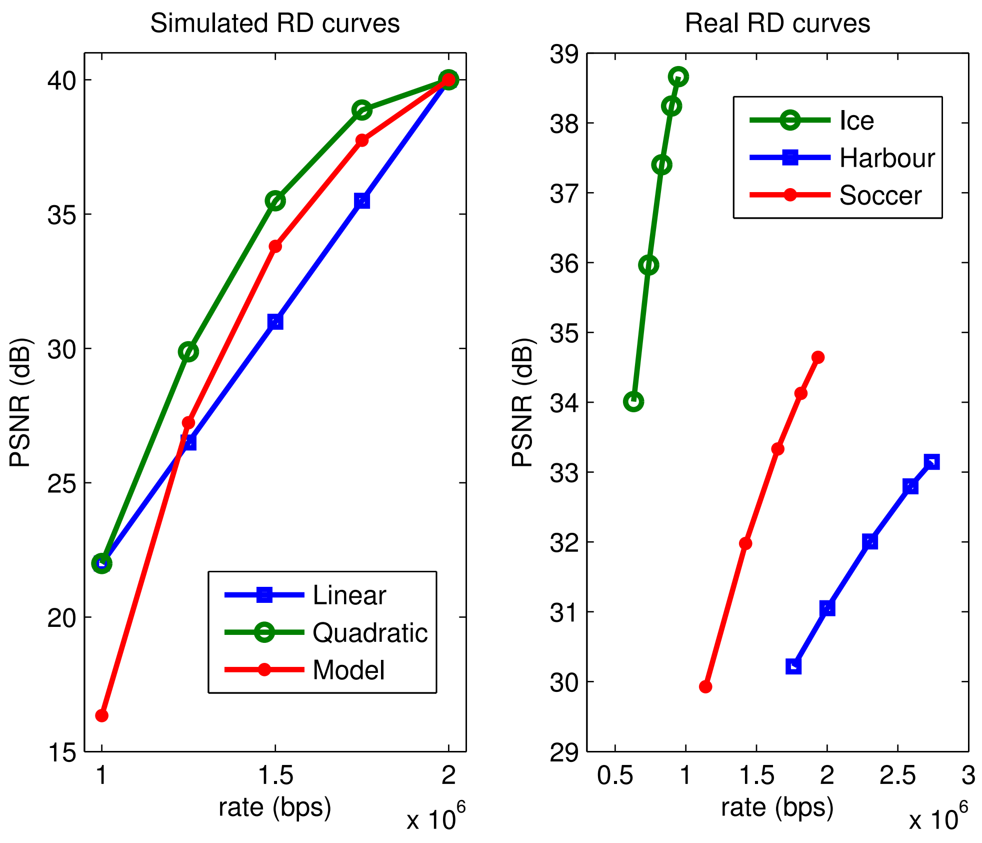


Figure 3.10: All RD curves used in simulations

slightly different from one another in terms of their adaptable PSNR ranges which resulted in different steepness of the curves. For example, in the linear simulation set, all users had linear RD curves but with different slopes. In addition, another simulation set was done with real RD curves from SVC-encoded Ice, Harbour and Soccer video clips at 4CIF (576×704) resolution. These videos whose RD curves are shown in the right of Figure 3.10 also had five operating points for the MGS scalability, but different adaptable bitrate and PSNR ranges compared to those of the simulated RD curves.

The coordination gain, as shown in Figure 3.11, is simply the difference between the Q_{norm} from the CAS and the UAS simulations measured at the same P_{norm}. A positive gain means the overall video quality for CAS users is better than that of the UAS users at the same overall channel situation whereas a negative gain implies the opposite. The plots for the gain are omitted once the P_{norm} goes too low beyond any of the CAS's or UAS's "Overloaded" points since the testbed with the mathematical model of the LTE cannot accurately represent the actual Q_{norm} in the "Overloaded" region as previously discussed. From Figure 3.9, the "Overloaded" region of the UAS covers up to $P_{norm} \approx 0.65$ given the current bitrate range between one and two Mbps of the simulated RD curves, and thus is why the gain plots for simulated RD curves are not shown below this point. The gain plot

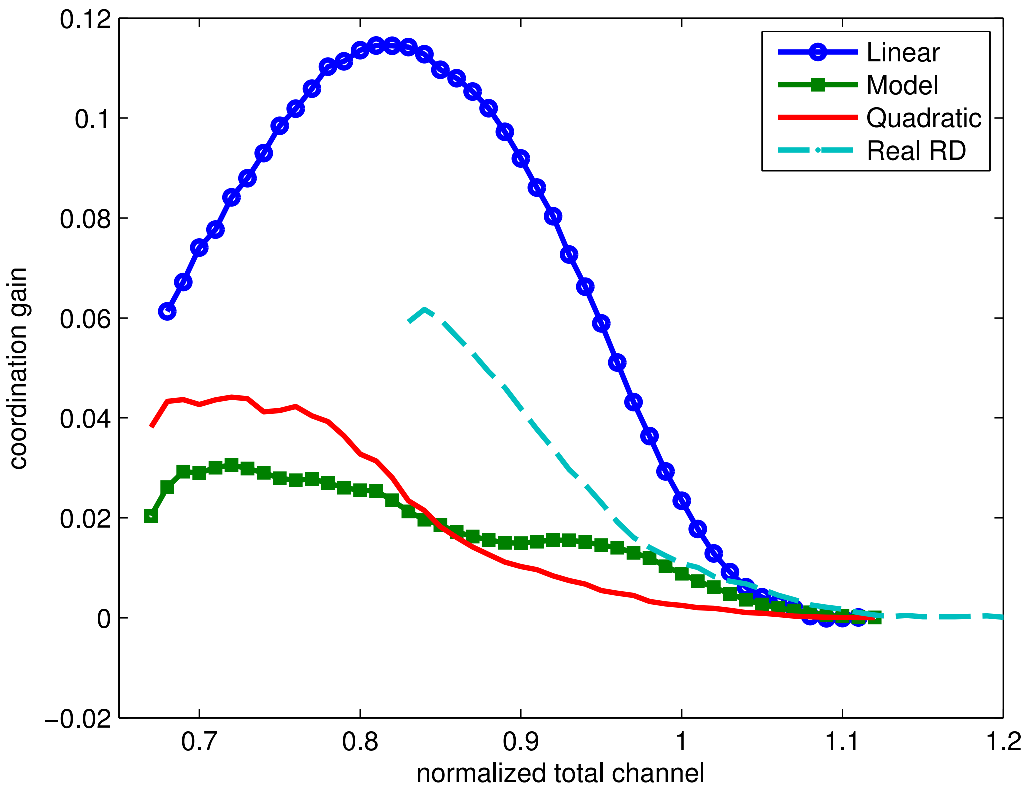


Figure 3.11: Coordination gain with various RD characteristics

for real RD curves, however, is omitted below a relatively higher P_{norm} since they have narrower adaptable bitrate ranges. Note that had the unadaptable AVC videos with zero adaptable bitrate range been included here for comparison, their Overloaded point would be close to one and the gain would always be zero beyond such point.

Figure 3.11 shows that once P_{norm} degrades below one into the Adaptive region, the coordination gain becomes positive, reaches its peak and declines before entering the Overloaded region where the users would be receiving only their base layers and possibly suffering losses. Interestingly, the gain tends to be higher when there are more linear RD curves in the system, e.g., the simulations with all linear RD curves and the real RD curves where some of them closely resemble a linear function. This is because the CAS's $\vec{C}_{opt}$ in this case is likely to be at the edge of $\widehat{E}_d$ as discussed in Section 3.3 whereas the $\vec{C}_{UAS}$ can be anywhere on $\widehat{E}_d$ depending on the channels. Thus, the Euclidean distance between the $\vec{C}_{opt}$ and the $\vec{C}_{UAS}$ when linear RD curves exist tends to be larger, and so does the difference in Q_{norm}, than having no linear RD curves and the $\vec{C}_{opt}$ is somewhere within the inner area of the $\widehat{E}_d$ plane.

Nevertheless, the coordination gains from these selected scenarios are rather subtle, e.g.,

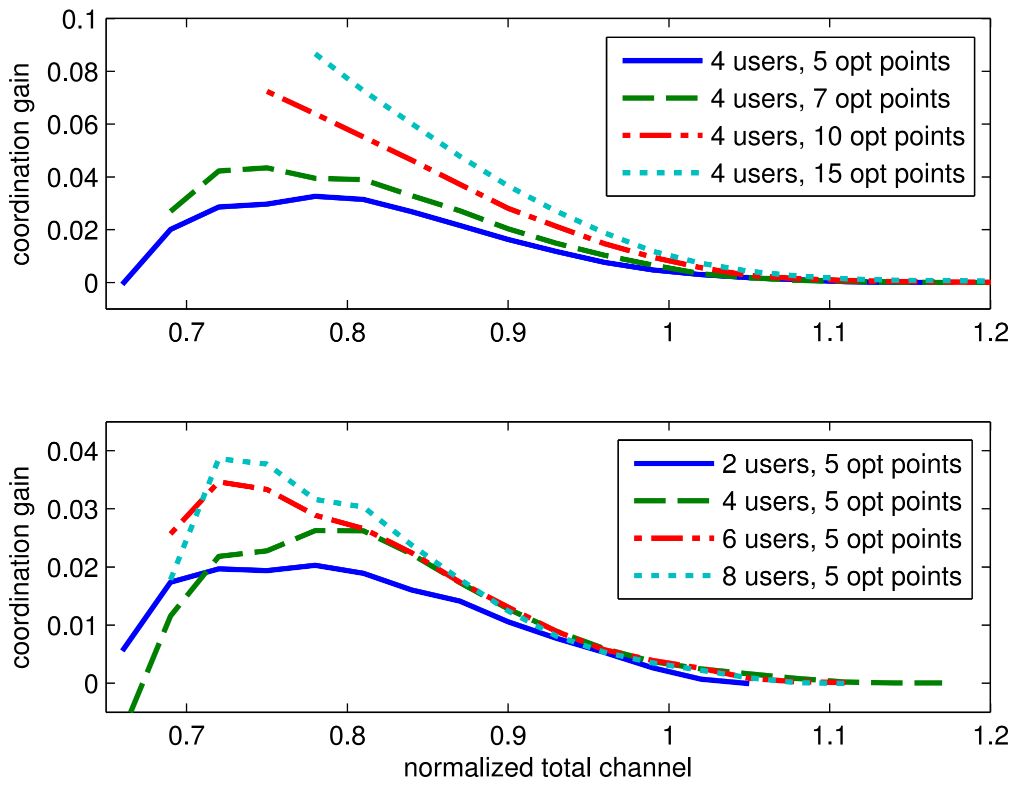


Figure 3.12: Coordination gain with different numbers of users and operating points

approximately 12% for the all-linear case at its peak. This translates into roughly a 1 dB increase in average PSNR for each user if the adaptable PSNR range is 10 dB.

3.6.3 The Number of Users and Operating Points

Apart from the characteristics of the RD curves, the number of users and operating points also influences the coordination gain. This section discusses results from simulations designed to demonstrate such effects. Simulations using the same quadratic RD curves in Section 3.6.2 with different slopes were performed for both CAS and UAS in two sets. In the first set, there were four streaming users in the cell with varying number of operating points between five to 15 each time. The plots of coordination gain versus the normalized total channel are shown in the upper part of the Figure 3.12. Similarly, each streaming user had five operating points on the RD curve but the number of users was varied from two to eight in the second simulation set. The lower part of Figure 3.12 shows the corresponding coordination gain of this simulation set.

It is clear from these results that increasing the number of users and operating points both improve the coordination gain significantly. Having more streaming users means the N-

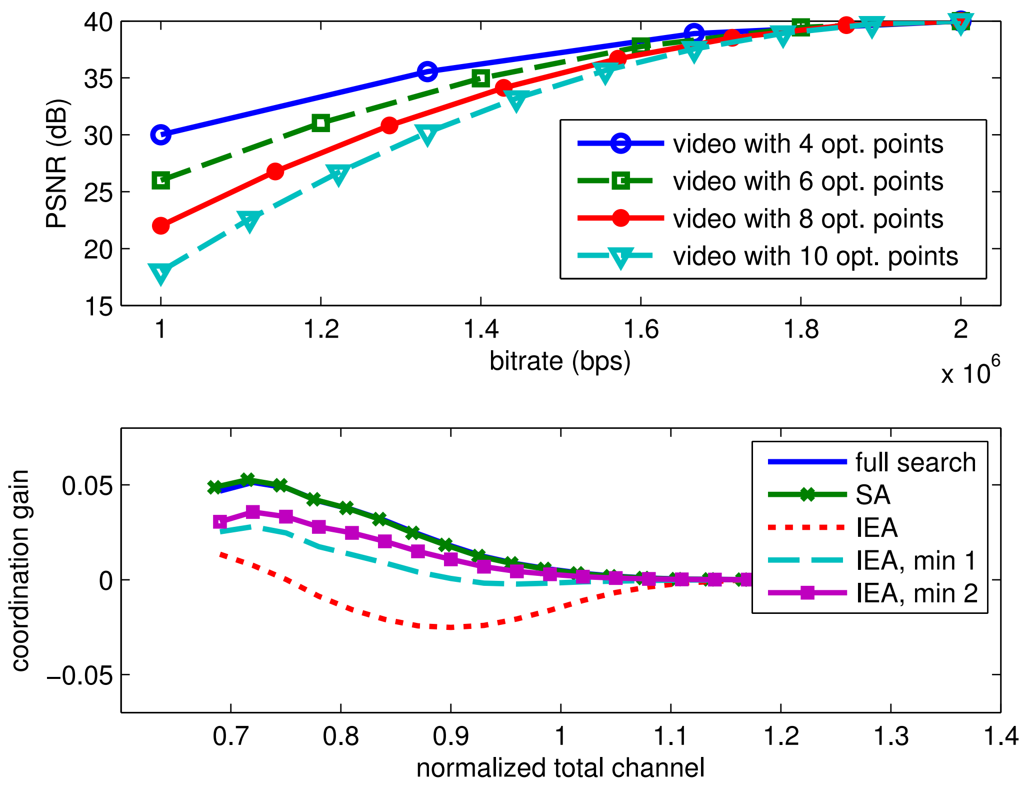


Figure 3.13: The RD curves and the coordination gains for different algorithms with more layers for steeper RD curves

dimensional solution space is larger. The Euclidean distance between the $\vec{C}_{opt}$ and $\vec{C}_{UAS}$ also tends to get larger as well as the difference in Q_{norm}, and therefore, the coordination gain. Similarly, having more operating points on the RD curves means the curves are being less discrete and more continuous which is the ideal optimum case already discussed in Section 3.3.

3.6.4 The Search Algorithms

To investigate the performance of search algorithms discussed in Section 3.3.3, two simulation sets with four streaming users using quadratic RD curves with different slopes were conducted. The number of operating points for four users varied from four to ten to mimic the situation in Figure 3.6 where the bitrate gaps between layers and users are significantly different. In one simulation set, the steeper RD curves shown in the upper part of Figure 3.13 were given more operating points, thus having smaller gaps. This increases the tendency that the SA algorithm is "distracted" by other ascending paths with steeper instantaneous slopes and smaller steps than to select the path with a more subtle slope

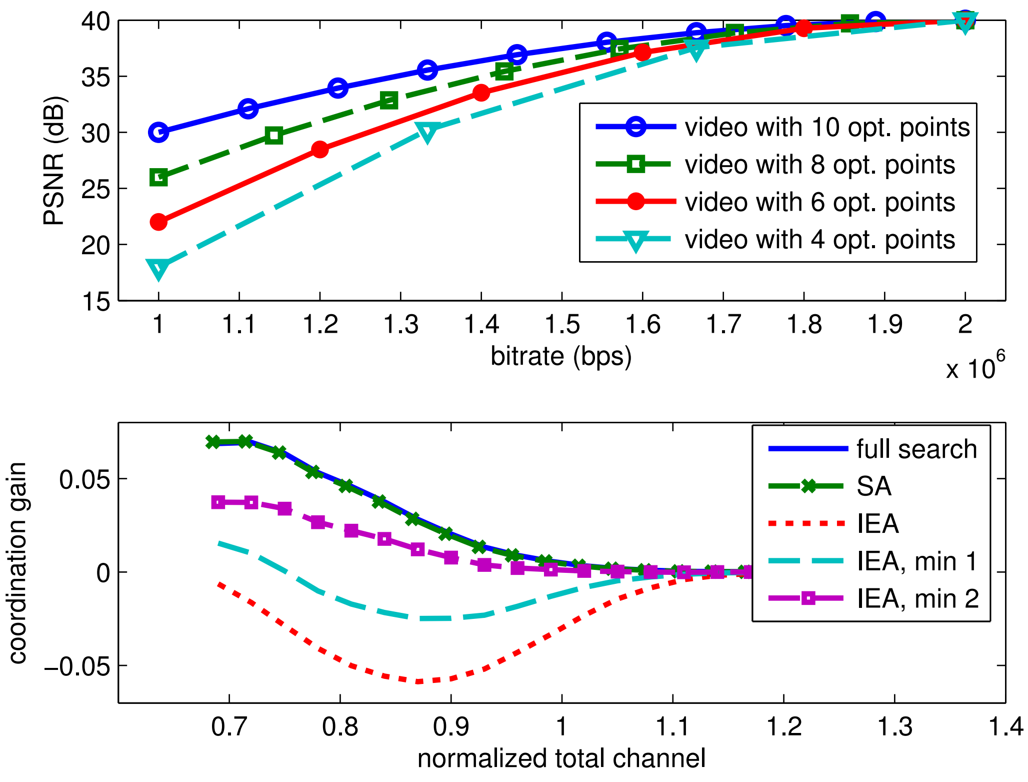


Figure 3.14: The RD curves and the coordination gains for different algorithms with fewer layers for steeper RD curves

but with a bigger step toward the true global optimum, as investigated earlier. The second simulation set on the contrary had RD curves with the number of layers in a reversed order as shown in the upper part of Figure 3.14.

The original discrete IEA algorithm, however, was expected to have difficulties in both scenarios. Thus, a modification has been made to it such that in each iteration, the least-affected user is downgraded enough such that another user which demands the smallest amount of resource can be upgraded by a layer. This improvement is referred to as "min 1". Similarly, a modified IEA labeled as "min 2" makes sure another user which demands the second smallest amount of resource can be upgraded by a layer at each iteration. The upgrading procedure is carried out using the SA algorithm at each iteration if the released resources are enough to upgrade multiple users and/or layers. These algorithms therefore have more adjacent solutions to compare with the current $\vec{C}^i$ in each round.

Simulation results reveal that the original IEA performs much worse than the others in both simulation sets including even the UAS, indicated by a negative coordination gain. This is because it is unable to move too far away from its initial random starting point in both scenarios. The modified IEA algorithms show better performance with the trade-off

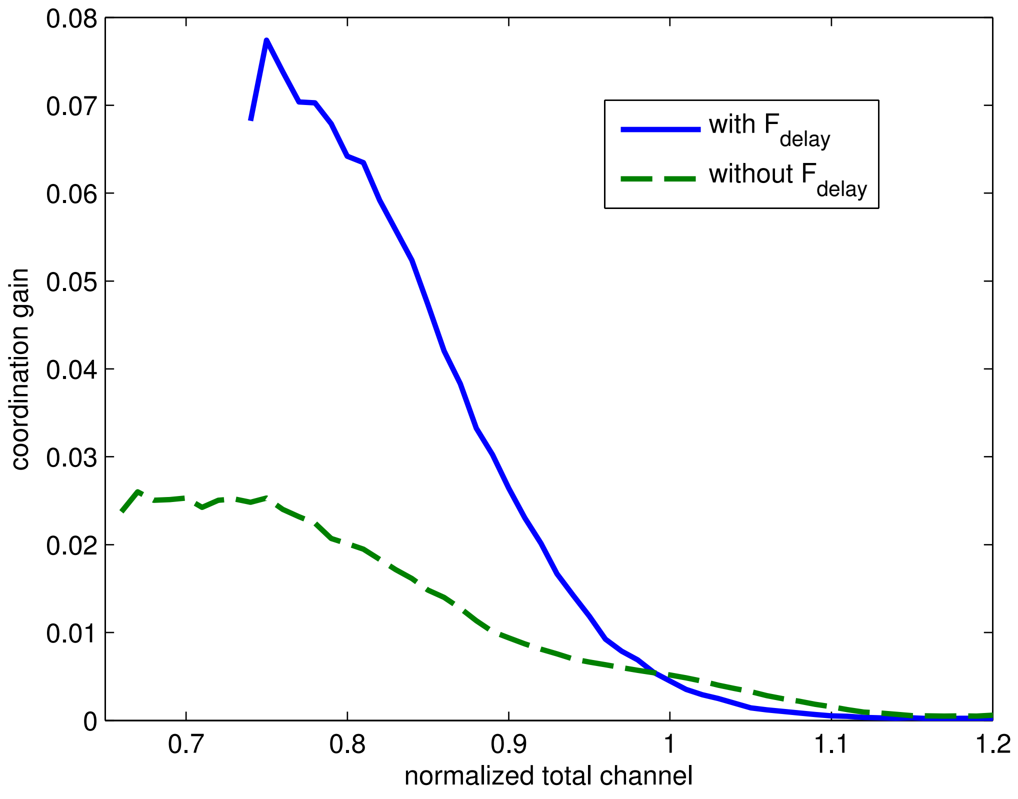


Figure 3.15: Coordination gain with and without the delay correction factor for the PF scheduler

in increased complexity. The performance of the discrete SA algorithm, on the other hand, closely matches that of the exhaustive search algorithm, implying its convergence to the $\vec{C}_{opt}$ most of the time. Interestingly the magnitude of the coordination gain (or loss) is also dependent on which RD curves get a larger number of layers as well.

3.6.5 The Scheduler's Sensitivity to Delay

To demonstrate the effects from the scheduler's sensitivity to the queuing delay on the CAS's performance as discussed in Section 3.3.4, the following simulations were conducted. In each simulation for CAS and UAS, there were four streaming users using four different quadratic RD curves with ten operating points. Both simulations are collectively referred to as one simulation set. In total, two simulation sets were done with the PF scheduler used at the base station. One simulation set was done with the F_{delay} in (2.1) set to one, representing a scheduler that is not sensitive to the delay. Another simulation set was done with the F_{delay} that grows exponentially with the queuing delay, thus increasing the metric value and the chance of a user with a long delay to be given more resources. The resulting coordination gain is shown in Figure 3.15 with an obvious quality improvement in favor of

Simulation length	20 minutes
Initial buffering period	7 seconds
System's bandwidth	5 MHz
User's speed	3 km/h
Cell's radius	100 m.
Resource scheduler	Proportional Fair
Channel simulation	interference, slow and fast fading

Table 3.2: Fixed simulation parameters for adaptive TBS

the system with the delay-sensitive scheduler as in earlier analysis.

3.6.6 Comparison with Non-Adaptive and TFRC-Based Adaptive Streaming

The simulations in this section were designed to compare the performances between CAS, UAS, TFRC-based adaptive video streaming and non-adaptive video streaming (referred to as the AVC simulation). For adaptive streaming cases, the SVC-encoded videos at 4CIF resolution introduced in Section 3.6.2 were used, whereas non-adaptive AVC-encoded videos of the same materials with similar bitrates were used for the AVC case. To simulate packet losses and the respective degradation to the video quality, the real-time LTE simulator described in Section 3.5.1 was used. There were three video streaming users and between zero to two other cross-traffic users, each generated a constant bitrate traffic at 500 Kbps to simulate various congestion levels in the cell. The GBR setting for a user was set according to the average bitrate of the requested video, or to 500 Kbps in the case of a cross-traffic user. Other fixed simulation parameters are as given in Table 3.2.

Figure 3.16 shows average PSNR and percentage of frozen frames as a result from the error-concealing mechanism due to losses versus a varying amount of traffic in the cell. These results agree with the expected relative performances between CAS, UAS and AVC cases. Specifically, the CAS which has a wider "Adaptive" region as discussed in Section 3.6.1 is more robust against increasing congestion, thus suffering fewer losses than in UAS and AVC cases. Therefore, the amount of repeated frames and the average PSNR of CAS users are better than the UAS and the AVC ones respectively. It is also slightly better than or comparable to those of the TFRC-based adaptation.

The Pause Intensity, I_p, in percentage for CAS, UAS and AVC cases as shown in the upper part of the Figure 3.17 are all zero across the simulation range. This is an expected behavior of a strictly timestamp-based streaming architecture as discussed in Section 2.3.1. However, the transmission rate from the TFRC-based adaptation server must adhere to the maximum allowed rate by the underlying DCCP protocol layer in which the TFRC rate control mechanism is built into. The DCCP layer can refuse to send data packets

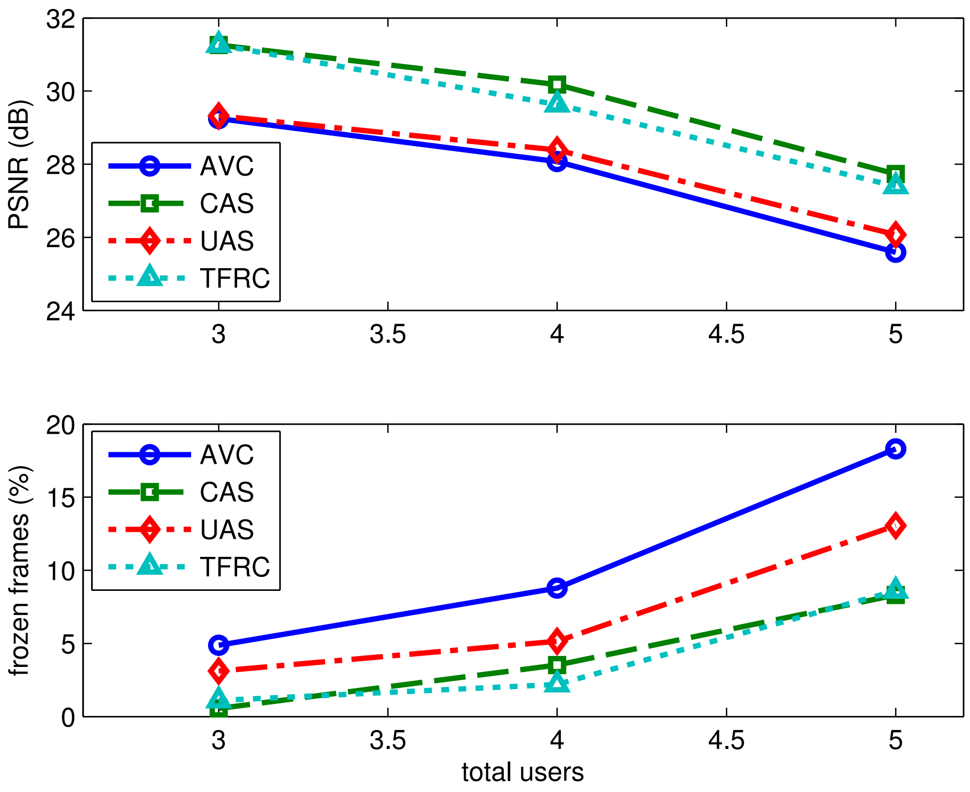


Figure 3.16: The average PSNR and amount of frozen frames for a streaming user at different traffic levels in the cell

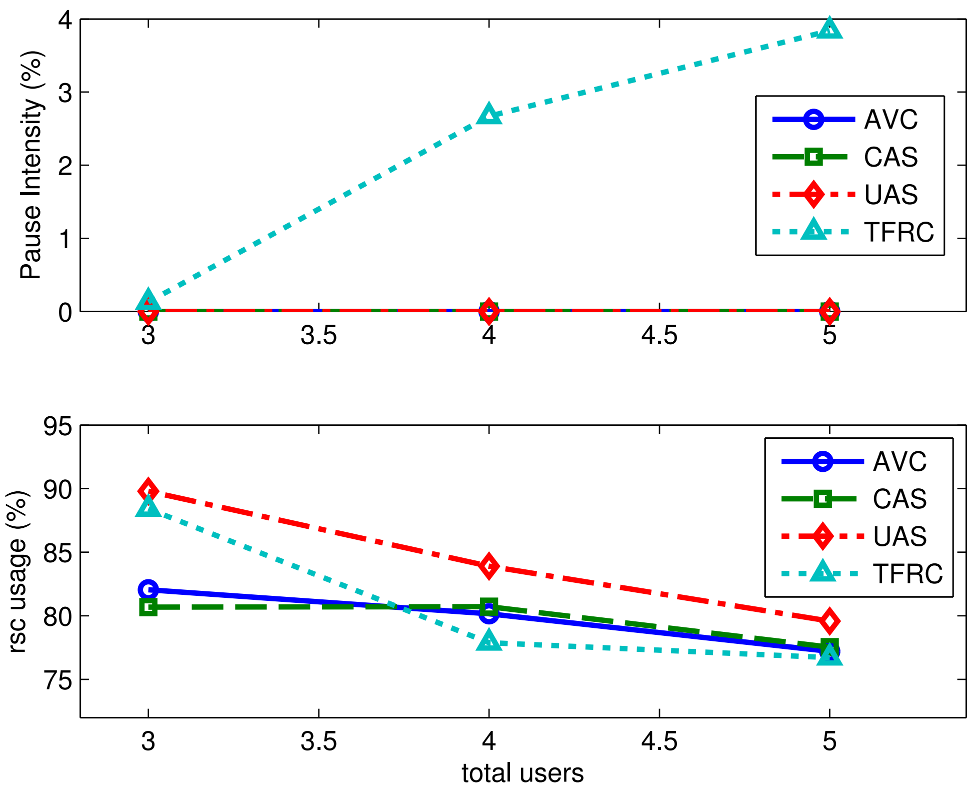


Figure 3.17: The Pause Intensity and radio resource usage for a streaming user at different traffic levels in the cell

through the network if the transmission rate would violate the rate deemed appropriate by the TFRC. This is equivalent to having an unbounded network delay T_N, thus making the B_{Rx} unable to be lower-bounded. It is therefore possible to have a non-zero I_p for the TFRC case, especially as the congestion in the cell increases as shown.

Finally, the lower part of the Figure 3.17 shows the average amount of radio resources used for streaming users in each case. This confirms that the fairness in resource allocation to other non-streaming users is preserved as the resource usage ratios for all adaptive cases are within 5-10 % of the AVC's. Additionally, note that the amount of used resources for CAS is mostly lower than the other cases although it has better performances as previously shown.

Chapter 4

Progressive-Download Adaptive Video Streaming

4.1 Motivation

With the increasing popularity of mobile video streaming and the expected high adoption rate by the industry of the DASH standard, a compatible client-based adaptive video streaming engine designed for the mobile environment will be a significant contribution for a successful deployment of such services in the near future. However, related works in this area and current state-of-the-art solutions discussed in Chapter 2 either only perform rudimentary adaptation or are incompatible with the standard. Additionally, knowledge regarding the statistical properties of TCP throughput over a mobile channel is still largely unexplored. Attempting to contribute to this missing crucial part, this chapter introduces the means to statistically estimate the TCP throughput in an OFDM-based mobile network, although theoretically it should still be applicable for W-CDMA based networks such as the HSPA as well. This knowledge is then used to construct a novel adaptation algorithm designed for the usage of SVC videos in the mobile environment by weighting the benefits and risks between different request strategies. Lastly, simulation results comparing the performance of the proposed algorithm with that of the state of the art in low and high-mobility scenarios are provided and analyzed.

4.2 Estimating Wireless TCP Throughput

This section introduces a statistical model to estimate the probability of getting a certain number of Bytes from a TCP connection over a wireless channel within given limited time duration. For a wireless channel based on the Carrier Sense Multiple Access (CSMA) technique, TCP is known to misinterpret channel losses as congestion-related losses and inappropriately reduces its transmission rate [87]. The conventional wisdom was therefore to enable the server to distinguish between congestion-related losses and channel losses so appropriate actions can be taken, e.g., to keep the transmission rate high but increase the FEC protection level for packets when the channel losses are detected. However, the LTE and other similar 3G/4G technologies have already included several retransmission mechanisms at their MAC and Physical layers to address such problem. Channel losses in these networks are converted into larger delay and congestion-related losses when the base station's buffer is full instead. Thus, the TCP's behavior of reducing its transmission rate on losses in modern mobile technologies is justified. The resulting TCP throughput therefore correctly represents the channel capacity without being interfered from other causes besides the network congestion. To gauge the potential TCP throughput over the mobile channel, the streaming application can then simply observe and deduce the channel situation from the long term throughput TCP provides.

In LTE, the allocation of radio resource chunks to mobile users is done by a base station's resource scheduler at every 1-2 ms Transmission Time Interval (TTI), as discussed in Section 2.1.2. Users with good instantaneous channels are likely to be given more radio chunks in successive TTI's to take advantage of their favorable channel conditions than others with worse channels, e.g, suffering deep fading. By measuring a series of instantaneous throughput values which reflect the channel and the scheduler's decision, the mobile user should be able to infer its future throughput as well. Let the estimated amount of Bytes the user gets in a period of T seconds be a random variable, denoted as β_T where T is in a range of 5-20 seconds. This can also be thought of as a sum of smaller random variables β_τ as well where $\tau << T$ to have a sufficient number of summands.

$$\beta_T = \beta_{\tau,1} + \beta_{\tau,2} + \ldots + \beta_{\tau,N} \; ; N = \lceil T/\tau \rceil \tag{4.1}$$

The underlying idea is that once some basic statistical properties of β_τ are known, the user should be able to use them to estimate the distribution of β_T. However, it is important not to set τ to be too small, especially to be comparable to the TTI period. This is because the TTI period is chosen to be smaller than the coherence time of the radio channel. Several successive TTI's tend to have similar channel conditions, either being under deep fade or having excellent reception quality altogether. The PF scheduler which assigns resource chunks based on the channel quality therefore either gives many resource chunks in several successive TTI's as a burst of data when the user is having a good reception quality or nothing at all in the opposite case. This results in most of the measured β_τ's having values close to zero and only a small fraction of them having large values representing a burst of data. Such β_τ show signs of having a heavy-tailed distribution with high degrees of correlation and a large or infinite variance, therefore making the Central Limit Theorem inapplicable to estimate β_T.

From experimental results, τ of 100 ms provides a good balance between having a sufficient number of β_τ samples and minimizing fluctuation and correlation among them since a β_τ

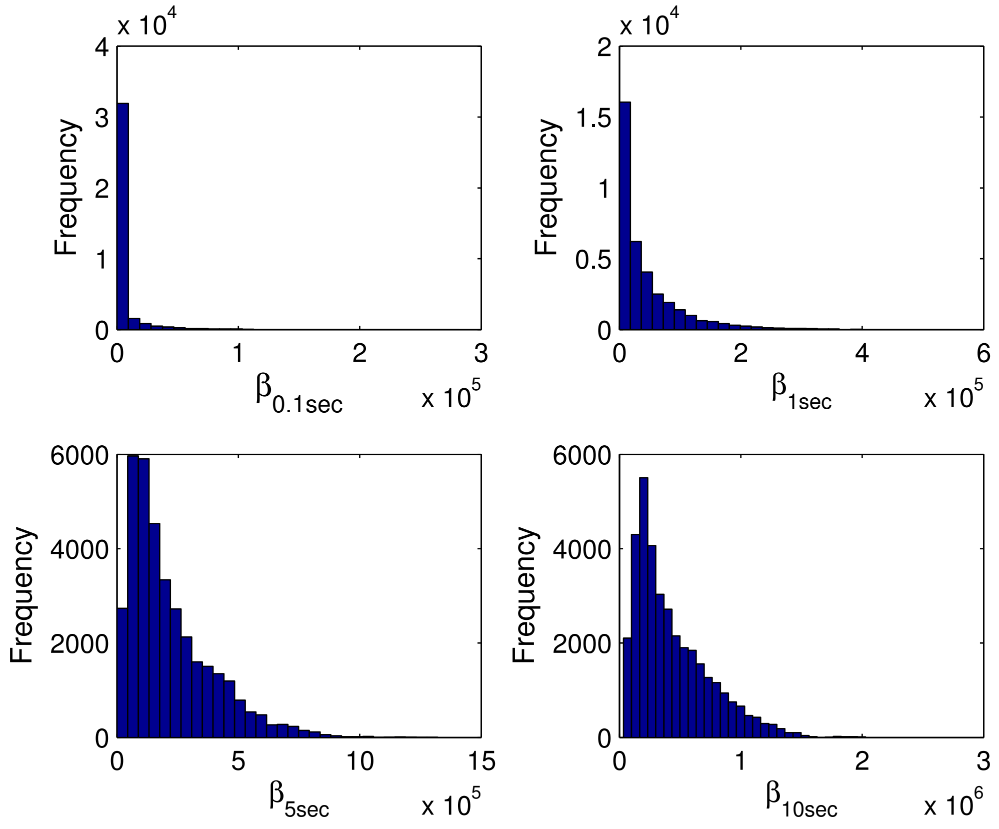


Figure 4.1: Distribution of β_T for different time periods

is a sum of all throughput contributions from as many as 50 or 100 individual TTI's. Thus short-term effects at the TTI level, e.g., having high correlation, are partially averaged, although signs that β_τ is heavy-tailed are still present as shown in the upper left histogram of Figure 4.1.

If β_τ is approximated by the heavy-tailed Log Normal distribution, β_T which is a sum of β_τ can also be approximated as Log Normal as well [88]. This is confirmed by the distribution plots of β_T with a long T period, e.g., β_{5sec} and β_{10sec} which are equivalent to a sum of 50 and 100 $\beta_{0.1sec}$'s respectively in the lower part of Figure 4.1. In such case, the tendency toward the Log Normal distribution of the distributions of the β_T where $\tau << T$ is still obvious. With these assumptions, $\beta_T \approx LogN\left(\mu, \sigma^2\right)$ where μ and σ^2 are the "location" and "scale" parameters that define the shape of the distribution. Using a similar approach as in [88], we first approximate them by pairing the mean of β_T with the mean of the sum as follows.

$$E\left(\beta_T\right) = E\left(\beta_{\tau,1} + \beta_{\tau,2} + \ldots + \beta_{\tau,N}\right) \tag{4.2}$$

$$e^{\mu+\sigma^2/2} = N \cdot E\left(\beta_\tau\right) \tag{4.3}$$

Similarly, pairing the variance of β_T and that of the sum yields the following.

$$var\left(\beta_T\right) = var\left(\beta_{\tau,1} + \beta_{\tau,2} + \ldots + \beta_{\tau,N}\right) \tag{4.4}$$

$$e^{2\left(\mu+\sigma^2\right)} - e^{2\mu+\sigma^2} = Nvar\left(\beta_\tau\right) + 2\sum_{i<j} Cov\left(\beta_{\tau,i}, \beta_{\tau,j}\right) \tag{4.5}$$

Solving (4.3) and (4.5), the μ and σ^2 can be written purely in terms of statistical properties of only β_τ as follows.

$$\mu = \ln\left[\frac{N^2E^2\left(\beta_\tau\right)}{\sqrt{Nvar\left(\beta_\tau\right) + 2\sum_{i<j} Cov\left(\beta_{\tau,i}, \beta_{\tau,j}\right) + N^2E^2\left(\beta_\tau\right)}}\right] \tag{4.6}$$

$$\sigma^2 = 2\left(\ln\left(N \cdot E\left(\beta_\tau\right)\right) - \mu\right) \tag{4.7}$$

Note that the covariance terms in (4.6) which involve all possible combinations of $\beta_{\tau,i}$ and $\beta_{\tau,j}$ where $1 \leq i < j \leq N$ are the result of breaking up the variance of the sum of correlated random variables β_τ in (4.4). These terms can be written in the expanded form as in the following.

$$\begin{aligned}
\sum_{i<j} Cov\left(\beta_{\tau,i}, \beta_{\tau,j}\right) &= Cov\left(\beta_{\tau,1}, \beta_{\tau,2}\right) + \\
&= Cov\left(\beta_{\tau,1}, \beta_{\tau,3}\right) + Cov\left(\beta_{\tau,2}, \beta_{\tau,3}\right) + \\
&= Cov\left(\beta_{\tau,1}, \beta_{\tau,4}\right) + Cov\left(\beta_{\tau,2}, \beta_{\tau,4}\right) + Cov\left(\beta_{\tau,3}, \beta_{\tau,4}\right) + \\
&\quad\vdots \\
&= Cov\left(\beta_{\tau,1}, \beta_{\tau,N}\right) + \cdots + Cov\left(\beta_{\tau,N-2}, \beta_{\tau,N}\right) + Cov\left(\beta_{\tau,N-1}, \beta_{\tau,N}\right)
\end{aligned} \tag{4.8}$$

In order to avoid calculating all these covariance terms, it is assumed that the correlation among $\beta_{\tau,i}$ and $\beta_{\tau,j}$ becomes negligible once $|i-j| > 2$, e.g, they are more than 200 ms apart when τ is chosen to be 100 ms. Let $\Phi_1 = Cov\left(\beta_{\tau,i}, \beta_{\tau,j}\right)$ where $|i-j| = 1$ and $\Phi_2 = Cov\left(\beta_{\tau,i}, \beta_{\tau,j}\right)$ where $|i-j| = 2$, we have the following approximation of these covariances.

$$\sum_{i<j} Cov\left(\beta_{\tau,i}, \beta_{\tau,j}\right) \approx \left(N-1\right)\Phi_1 + \left(N-2\right)\Phi_2 \tag{4.9}$$

T	$\upsilon = 0.1$	$\upsilon = 0.2$	$\ldots$	$\upsilon = 0.9$
T_1	b_{11}	b_{21}	$\ldots$	b_{91}
T_2	b_{12}	b_{22}	$\ldots$	b_{92}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
T_t	b_{1t}	b_{2t}	$\ldots$	b_{9t}

Table 4.1: An example look-up table of b_υ with respect to υ and T

Equation (4.9) greatly reduces the complexity of this step. With this approximation, the user is only required to compute two multiplication operations in each τ period i for Φ_1 and Φ_2, specifically the products between the newly observed $\beta_{\tau,i}$ and $\beta_{\tau,i-1}$ and between $\beta_{\tau,i}$ and $\beta_{\tau,i-2}$ respectively which are then kept in their own separated memory arrays. Whenever the values of Φ_1 and Φ_2 are needed, they can simply be obtained from the average values of the entries in these arrays as follows.

$$\Phi_1 = E\left(\beta_{\tau,i} \cdot \beta_{\tau,i-1}\right) - E^2\left(\beta_\tau\right) \tag{4.10}$$

$$\Phi_2 = E\left(\beta_{\tau,i} \cdot \beta_{\tau,i-2}\right) - E^2\left(\beta_\tau\right) \tag{4.11}$$

Note that it is possible to improve the estimation in (4.9) by adding $\Phi_3, \Phi_4, \ldots$ to it especially if τ is chosen to be smaller than 100 ms. However, this comes at the expense of increasing complexity.

From the array of recorded samples of β_τ and the earlier discussed arrays of products, the user can estimate the μ and σ^2 from (4.6) and (4.7) with the simplifications made in (4.9), (4.10) and (4.11). The success probability υ of getting b_υ Bytes within T seconds, denoted as $P\left(\beta_T \geq b_\upsilon, T\right)$ or $P\left(b_\upsilon, T\right)$ can then be derived from the Log Normal's Complementary CDF as

$$P\left(b_\upsilon, T\right) = \upsilon = \frac{1}{2} - \frac{1}{2} erf\left(\frac{\ln\left(b_\upsilon\right) - \mu}{\sqrt{2\sigma^2}}\right) \tag{4.12}$$

$$b_\upsilon = \exp\left(\sqrt{2\sigma^2} \cdot \chi_\upsilon + \mu\right) \tag{4.13}$$

Here, $\chi_\upsilon = erf^{-1}\left(1 - 2\upsilon\right)$ depends only on the success probability υ and can be pre-computed beforehand. A look-up table of b_υ with respect to various υ and T similar to Table 4.1 can be quickly constructed every time the streaming application needs to estimate the success probability. For example, the success probability υ to obtain b_υ Bytes through the channel within T_2 seconds is $0.1 < \upsilon < 0.2$ if $b_{22} < b_\upsilon < b_{12}$. Note that the relationship between T and N from (4.1) is $N = \lceil T/\tau \rceil$.

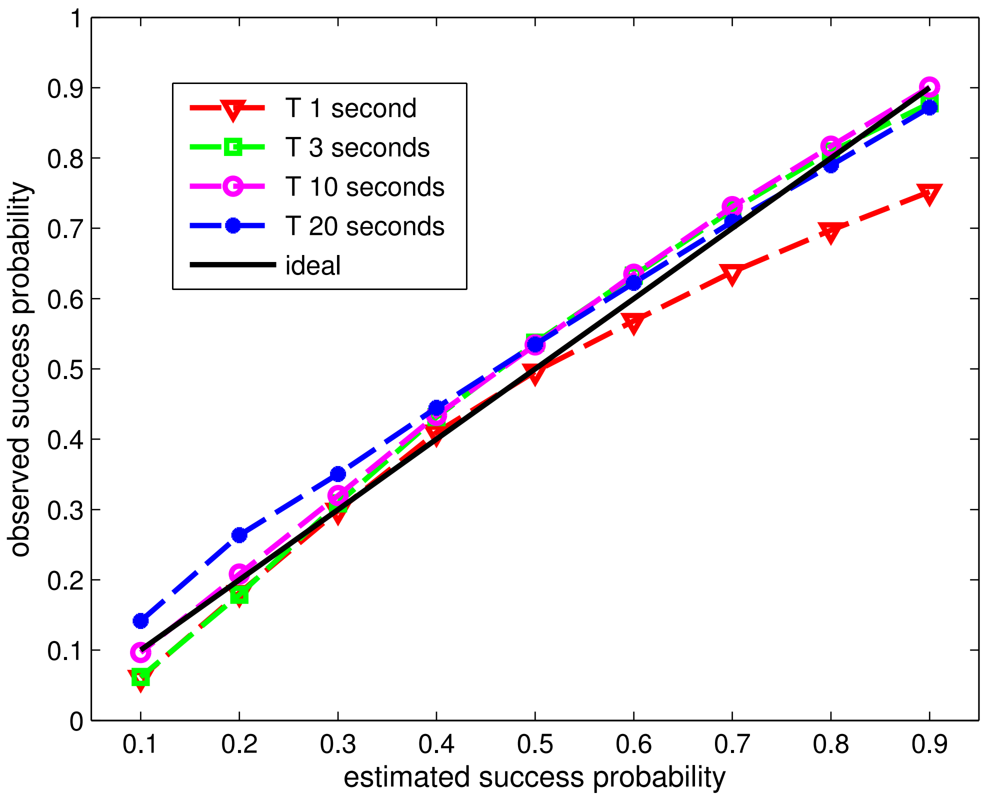


Figure 4.2: Estimated and actual success probabilities

Figure 4.2 shows the accuracy of the estimation from simulations where the estimated success probability v is plotted against the observed probability that the amount of Bytes received within T seconds is no less than the corresponding b_v. The result shows satisfactory accuracy where estimations for various values of T are most of the time very close to the "ideal" line. The settings for these simulations will be covered in more details later in Section 4.5.

4.3 Content Preparation

As discussed earlier in Section 2.3.2, both the AVC and SVC videos are usually prepared into smaller self-decodable units called "chunks" for usage with the PD paradigm. A chunk can be, e.g, a collection of one or more Group of Pictures (GoP) which is independent of other chunks. In this work, it is proposed that for SVC-encoded videos, smaller NAL units in each chunk are grouped further into "blocks" such that each block represents only a single scalable layer of that chunk. This is shown in Figure 4.3 where a chunk of one GoP with four temporal and one MGS layers is segmented into eight more blocks. Each block is given a layer ID from 0, representing the base layer, to $L-1$ where L is the total number

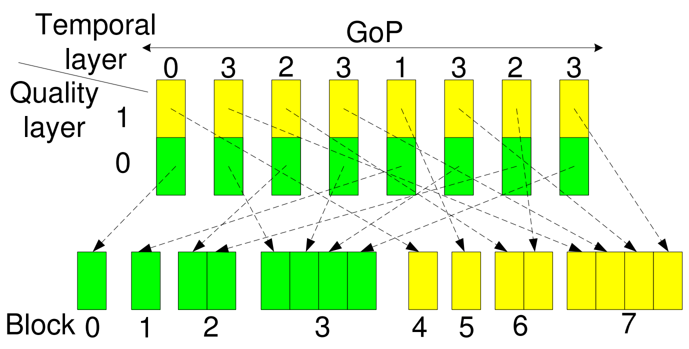


Figure 4.3: An example on preparing video content into blocks

of all scalable layers in the video. This means at least block 0 must be present in order to decode any given chunk. More enhancement blocks can be requested given that their "ancestry" blocks have all been requested before.

The hierarchical relationships between blocks of the same chunk are defined by the decoding dependency of each of them on the other blocks, e.g., the inter-layer prediction vectors, etc. It is also recommended that the encoding structure of the video should allow for RD-optimized adding/removing of scalable layers as well. As an example, temporal layer blocks should have lower indexes (more important) than MGS quality layer blocks due to the quality improvement in having a higher frame rate is generally better than having more MGS layers. In addition, the RD information in terms of quality improvement, e.g, $\Delta PSNR$ that each layer contributes from its lower layer of each chunk must be generated during encoding and assumed to be made available to the user as well.

4.4 Decision Algorithm

4.4.1 Definitions

Let each block from a video with L layers be identified by a coordinate pair (c, l) where $c \in \{0, 1, \ldots\}$ and $l \in \{0, 1, \ldots, L-1\}$ denote the chunk and layer ID respectively. In addition, let $\Pi(c, l)$ be the size of the block and $\Delta Q(c, l)$ be the quality improvement, e.g., $\Delta PSNR$, from its lower-layer block $(c, l-1)$ if $l > 0$ or the total quality if $l = 0$. Let T_p be the chunk duration, these new definitions are related to the definitions of the bitrate and quality of a chunk c at layer l given earlier in Section 3.2 which are denoted as $R_{c,l}$ and $Qlt_c(R_{c,l})$ respectively as follows.

$$R_{c,l} = \frac{\sum_{i=0}^{l} \Pi(c, i)}{T_p} \tag{4.14}$$

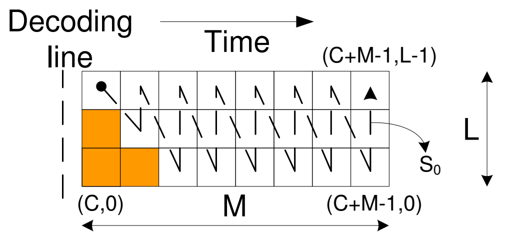


Figure 4.4: An adaptation window of size $M \times L$

$$Qlt_c\left(R_{c,l}\right) = \sum_{i=0}^{l} \Delta Q\left(c,i\right) \tag{4.15}$$

The user manages its receiving buffer by constructing an imaginary "adaptation window" of size $M \times L$ blocks as shown in Figure 4.4, implying enough memory space to store M chunks in advance. The "decoding line" marks the chunk currently being retrieved and decoded from the buffer. This window is shifted forward continuously such that it always covers the next M chunks from the decoding line. In addition, let the simulation time t start from 0 and the initial buffering time be T_{init}. Thus the time remaining for a block (c,l) in the window from the current simulation time until it is due to be decoded is

$$T_R\left(\left(c,l\right),t\right) = T_{init} + cT_p - t \tag{4.16}$$

Let K be the number of unrequested blocks remaining in the adaptation window. A request sequence $\widehat{S}$ is defined as follows.

$$\widehat{S} = \left\{ (c,l)_i \,|\, (c,l)_i \text{ is missing}, \left((c,l)_j \prec (c,l)_i\,, (c,l)_j \text{ is missing} \right) \rightarrow (1 \leq j < i \leq K) \right\} \tag{4.17}$$

Verbally, this is a set of all K missing blocks in the adaptation window and represents the schedule the user will request for these blocks from the server sequentially based on the order of their appearances in the set, denoted by a subscript i. The ordering of blocks in $\widehat{S}$ must also comply with the dependencies between them. For example, any missing ancestry blocks $(c,l)_j$ of the block $(c,l)_i$, denoted by $(c,l)_j \prec (c,l)_i$, must be requested earlier in the sequence $\widehat{S}$, e.g., $1 \leq j < i \leq K$. Some examples of very conservative and aggressive request sequences, assuming the adaptation window is completely empty, are given below.

$$\begin{aligned}\widehat{S}_{cons} = \{&(C,0),(C+1,0),\ldots(C+M-1,0),\\ &(C,1),(C+1,1),\ldots(C+M-1,1),\\ &\vdots\\ &(C,L-1),(C+1,L-1),\ldots(C+M-1,L-1)\}\end{aligned} \tag{4.18}$$

$$\begin{aligned}\widehat{S}_{aggr} = \{&(C,0),(C,1),\ldots(C,L-1),\\ &(C+1,0),(C+1,1),\ldots(C+1,L-1),\\ &\vdots\\ &(C+M-1,0),(C+M-1,1),\ldots(C+M-1,L-1)\}\end{aligned} \tag{4.19}$$

In each decision instance, there can be many possible request sequences the user can choose from. A total quality metric for any sequence $\widehat{S}$ at time t is therefore defined in order to make a comparison amongst them as follows.

$$\Delta Q_{tot}\left(\widehat{S},t\right) = \sum_{i=1}^{K}\left[\Delta Q\left((c,l)_i\right) P\left(\Pi_i, T_R\left((c,l)_i,t\right)\right)\right] \tag{4.20}$$

Each individual summand in (4.20) is the quality contribution $\Delta Q\left((c,l)_i\right)$ of each block $(c,l)_i$ in $\widehat{S}$ weighted by its success probability in getting all the preceding $i-1$ blocks and itself within its decoding time as defined in (4.12) and (4.16) given the current estimated channel conditions. Here, Π_i represents the total size of the first i blocks in $\widehat{S}$, e.g, $\Pi_i = \sum_{j=1}^{i}\Pi\left((c,l)_j\right)$. Finally, the success probability to weight each term is obtained from a look-up table similar to Table 4.1 with M rows representing different decoding times of all M chunks in the adaptation window. This allows the streaming application to determine the best request sequence with the largest ΔQ_{tot} and request for blocks in the same order as appear in the sequence.

4.4.2 Determining the Best Request Sequence

In this section, a sub-optimum algorithm to determine the best request sequence, referred to as $\widehat{S}_{opt}$, is introduced. This is necessary since obviously the user cannot make a brute-force search through all possible request sequences whenever it needs to make an adaptation decision in real time. Define and initialize $\widehat{R}$ as a set blocks that have already been received and stored in the adaptation window. Also, let $\widehat{I}_{\widehat{R}}$ be a set of all blocks that can be immediately requested given that all of their ancestry blocks have been included in $\widehat{R}$.

For example, the shaded blocks in Figure 4.4 which represent the already-received blocks in the window are included in $\widehat{R}$ at initialization. The $\widehat{I}_{\widehat{R}}$ in this case is initialized to include blocks $(C, 2)$, $(C + 1, 1)$ and $(C + 2, 0)$ since all of their ancestry blocks are already present in $\widehat{R}$.

The concept of the proposed algorithm is to solve the multivariate optimization problem of finding $\widehat{S}_{opt}$ iteratively by simplifying it into an optimization problem with a single variable in each iteration round. This can be achieved by initializing any “starting sequence” of the search first. Then design an algorithm to determine the best block to request amongst all applicable candidate blocks at each position while considering other positions as fixed. This process is repeated iteratively, starting from the beginning to the end of the sequence. Once the algorithm terminates, the starting sequence will have been modified into a better optimized solution.

The choice of the starting sequence can be anything that adheres to the restrictions given in (4.17). However, experimental results have shown that by using a sequence similar to either $\widehat{S}_{cons}$ in (4.18) or $\widehat{S}_{aggr}$ in (4.19) as the starting sequence appropriately, depending on the channel situation, the adaptation performance is noticeably improved. Specifically, $\widehat{S}_{cons} - \widehat{R}$ should be used as the starting sequence in the bad channel situation as it tries to get only the missing base-layer blocks first to prevent frequent playback interruptions. It is therefore intuitive that $\widehat{S}_{opt}$ in such situation should already be similar to $\widehat{S}_{cons} - \widehat{R}$ and require fewer “modifications” by the algorithm. By initializing the starting point of the search to be closer to $\widehat{S}_{opt}$, the algorithm is less likely to be distracted to other local sub-optimum solutions and thus the adaptation result is improved. The same reasoning also applies for the opposite scenario of using $\widehat{S}_{aggr} - \widehat{R}$ as the starting sequence when the user is in a good reception area as well. Let $\tilde{R}$ be the past average throughput observed by the user, the starting sequence $\widehat{S}_0$ can therefore be chosen in relation to the time-averaged video bitrate at the base layer as follows.

$$\widehat{S}_0 = \begin{cases} \widehat{S}_{cons} - \widehat{R} & \text{if } \tilde{R} \le \epsilon \cdot mean\left(R_{c,0}\right) \\ \widehat{S}_{aggr} - \widehat{R} & \text{if } \tilde{R} > \epsilon \cdot mean\left(R_{c,0}\right) \end{cases} \tag{4.21}$$

The multiplier $\epsilon > 0$ is used to control how conservative the algorithm should be. From experimental results, setting $\epsilon \approx 1.5$ proves to be a good balance between minimizing the chances of interruption and having fast responsiveness to improving channel situations.

Once $\widehat{S}_0$ has been selected, the proposed Algorithm 1 is executed. At each step $n = 1, \ldots, K$, the best block to request at the n^{th} position of $\widehat{S}_{n-1}$ from the previous step is selected out of all the blocks that can be immediately requested at this position, which by definition are the members of $\widehat{I}_{\widehat{R}}$. Denote these blocks as $(c, l)_p \in \widehat{I}_{\widehat{R}}$ where $p = 1, 2, \ldots, \left|\widehat{I}_{\widehat{R}}\right|$ and $\left|\widehat{I}_{\widehat{R}}\right|$ is the total number of blocks in $\widehat{I}_{\widehat{R}}$. For each p, the algorithm tries moving a block $(c, l)_p$ from its original position to the n^{th} position of $\widehat{S}_{n-1}$ instead, denoted the modified

```
for $n = 1 : K$ do
    foreach $(c,l)_p \in \widehat{I}_{\widehat{R}}$ where $p = 1, 2, \ldots, \left|\widehat{I}_{\widehat{R}}\right|$ do
        $\widehat{S}_n^p = \widehat{S}_{n-1}$;
        Move $(c,l)_p$ from its original location to the $n^{th}$ location in $\widehat{S}_n^p$;
        Calculate $\Delta Q_n^p\left(\widehat{S}_n^p, t\right)$;
    end
    $\widehat{S}_n = \arg\max_{\forall \widehat{S}_n^p}\left(\Delta Q_n^p\left(\widehat{S}_n^p, t\right)\right)$;
    Update $\widehat{R} = \widehat{R} \cup \{(c,l)_n\}$ where $(c,l)_n \in \widehat{S}_n$;
    Update $\widehat{I}_{\widehat{R}}$;
end
```

Algorithm 1: The algorithm to determine $\widehat{S}_{opt}$

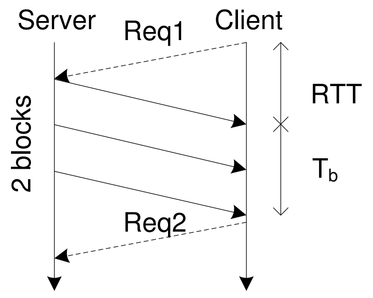


Figure 4.5: The RTT measurement by the user

sequence as $\widehat{S}_n^p$. The best block to be placed at this location, or equivalently the best modified sequence out of all $\left|\widehat{I}_{\widehat{R}}\right|$ possible modified sequences at this iteration round is the one that yields the highest metric in (4.20) and is selected to be $\widehat{S}_n$. The selected block at this iteration round is also added to $\widehat{R}$, and $\widehat{I}_{\widehat{R}}$ has to be updated for the next round accordingly. These steps continue until $n = K$ and the blocks to request at all K locations are selected, thus $\widehat{S}_{opt} = \widehat{S}_K$ is found.

However, the streaming user should not request all missing K blocks in $\widehat{S}_{opt}$ at once since the transmission time required for all these blocks could last several seconds, during which time, the channel might have changed and a better adaptation decision could and should be made instead. On the other hand, requesting too few blocks from $\widehat{S}_{opt}$ would result in low downlink channel utilization, especially if the RTT is relatively large compared to the transmission time of a block. Thus, the amount of blocks, or Bytes to request is determined such that the approximated channel utilization exceeds a certain desired threshold, e.g, more than 80% utilization rate. This requires the user to take regular measurements of the

RTT delay between sending the request until the first Byte of the reply arrives as shown in Figure 4.5. The utilization rate U where T_b is the transmission time for all the requested blocks is

$$U = T_b / \left(T_b + RTT\right) \tag{4.22}$$

Therefore, the minimum number of Bytes the user must request such that the target minimum utilization rate U_{min} is achieved can be derived as follows.

$$B_{min} = (\text{minimum Tx period to achieve } U_{min})(\text{avg. throughput}) \tag{4.23}$$

$$B_{min} = \left(\frac{U_{min} \cdot RTT}{1 - U_{min}}\right) \cdot \left(\frac{E\left(B_\tau\right)}{\tau}\right) \tag{4.24}$$

4.4.3 Summary of the Algorithm

In summary, the procedures the streaming application must do can be classified into those that are performed when the user is receiving video blocks from the server and those that are performed when the user needs to make the next adaptation decision (once all the previously requested blocks have been received).

Procedures during Receiving Data

- Measure the RTT delay between sending of the HTTP request message and receiving of the reply from the server as demonstrated in Figure 4.5.
- In each period i which lasts for τ seconds, record the amount of Bytes received within that period as $\beta_{\tau,i}$ and store it in an array. Refer to this as the array I_0. Similarly, store the products $\beta_{\tau,i} \cdot \beta_{\tau,i-1}$ and $\beta_{\tau,i} \cdot \beta_{\tau,i-2}$ in arrays I_1 and I_2 respectively.

Procedures to Make Adaptation Decision

- Calculate the respective average values of the entries in arrays I_0, I_1 and I_2. Then, clear all arrays.
- Use the average values from the three arrays to calculate Φ_1 and Φ_2 according to (4.10) and (4.11) respectively.
- Construct a look-up table similar to Table 4.1 with M rows, each corresponding to M different decoding deadlines of the chunks in the adaptation window as given in (4.16). The entries in each row can be computed from (4.6), (4.7) and (4.13).

Simulation length	128 minutes
Initial buffering period	7 seconds
System's bandwidth	5 MHz
User's speed	3 or 30 km/h
Cell's radius	100 m.
Resource scheduler	Proportional Fair
Channel simulation	interference, slow and fast fading

Table 4.2: Fixed simulation parameters for PD adaptive streaming

- Select the starting sequence $\widehat{S}_0$ in (4.21).
- Compute the amount of Bytes to request using (4.24).
- Execute Algorithm 1 to determine the $\widehat{S}_{opt}$. Note that the algorithm can be further simplified even more since only B_{min} Bytes will be requested from the server anyways. This implies that not all blocks in $\widehat{S}_{opt}$ will actually be requested at this time once the algorithm finishes. Therefore, the algorithm can be terminated prematurely without having to iterate through all K positions in the sequence. Instead, it can be terminated at any iteration round n as soon as the accumulated size in Bytes of the first n blocks in $\widehat{S}_n$ exceeds B_{min}.
- Send HTTP requests for the first n blocks in $\widehat{S}_n$. Start the timer again to measure the RTT.

4.5 Simulation Results and Analyses

4.5.1 Simulation Testbed

Simulations to compare the performances between the proposed adaptive algorithm with a SVC-encoded content, a typical state-of-the-art adaptive streaming algorithm with multiple representations of the non-scalable AVC-encoded content, and a simple non-adaptive streaming with an AVC-encoded content were conducted. These are referred to as the adaptive SVC, adaptive AVC and non-adaptive AVC from now on respectively. The simulation testbed as shown in Figure 4.7 was used to conduct these simulations where the LTE simulator module represents the real-time simulator for a LTE cell [22]. The HTTP server module represents a typical HTTP server storing the video blocks somewhere on the Internet. Finally the User module represents a single video streaming user in the LTE cell which can be operated in all three modes. However, there were also between three to nine other cross-traffic users in the cell internally generated by the simulator as well to represent various congestion levels. All users are served in a best-effort manner with no QoS guarantee. Other general testbed configurations are as given in Table 4.2.

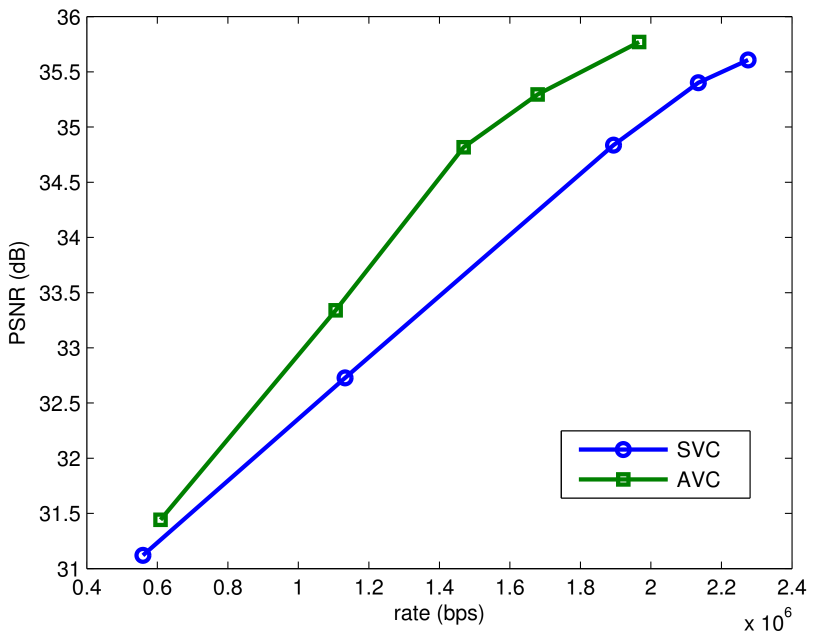


Figure 4.6: RD curves for videos used in the simulations

The video used in the simulation is a concatenation of shorter Crew, Mobile and Soccer video clips. It is further encoded into an SVC version with five scalable layers and five AVC representations with similar bitrate ranges. The base layer of the SVC video has the CIF resolution and the remaining four layers are spatial and MGS scalable layers, all at 4CIF resolution. Similarly, the first AVC representation is at the CIF resolution while the remaining four are at 4CIF resolution with different QP values. There are two GoP's with 32 frames in total in each video chunk. The playback rate is at 30 fps, thus resulting in a chunk duration of 32/30 seconds. The RD curves for both AVC and SVC videos with all their operating points are shown in Figure 4.6.

The behavior of the video playback is assumed to be as for any other typical HTTP streaming application. The playback starts after an initial buffering period once the first data packet has arrived. If the user's buffer becomes empty anytime during the playback, the player is forced to stop. It resumes the playback again as soon as there is enough data in the buffer to do so. Additionally, specific configurations for the proposed adaptive

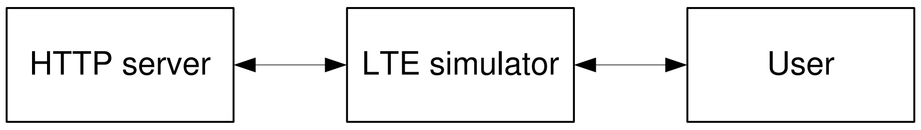


Figure 4.7: A diagram for the PD adaptive streaming testbed's components

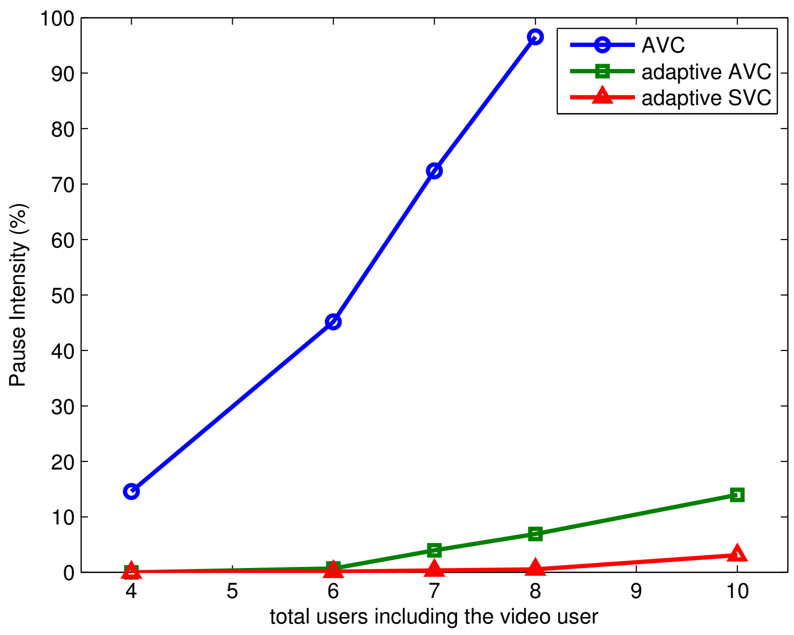


Figure 4.8: Pause intensity at different congestion levels for low-mobility scenario

algorithm are as follows. The adaptation window is of size 60×5, $\tau = 100$ ms and finally the user keeps the β_τ entries from the last 10 seconds in the three arrays earlier discussed.

Regarding the adaptation algorithm of the adaptive-AVC simulations, it is designed to mimic general behaviors of the current state-of-the-art algorithms available commercially. Specifically, the user requests for video chunks sequentially where the chosen representation for each chunk is the one with the required bitrate no larger than the past average bitrate. If the receiving buffer is full, e.g., all chunks in its adaptation window of size $M \times 1$ have been requested, the algorithm additionally tries to improve the quality of the undecoded chunks in the window by requesting better representations for them as well.

There are two main scenarios investigated in this work, the low-mobility scenario where all users are moving randomly in the cell at three km/h, and the high-mobility scenario where the users are moving at vehicular speed of 30 km/h. The latter scenario is especially of interest since the statistical model of the best-effort TCP throughput used to predict the success probability was originally construct by analyzing channel traces at low speed which might be less accurate when used in the high-speed environment. Thus, it is interesting whether the model and the algorithm can still perform as well in this case or not.

4.5.2 Low-Mobility Scenario

In this section, the results and analyses for the low-mobility simulations are presented. The plots for the Pause Intensity as defined in Section 2.2.3 in percentage versus traffic load in

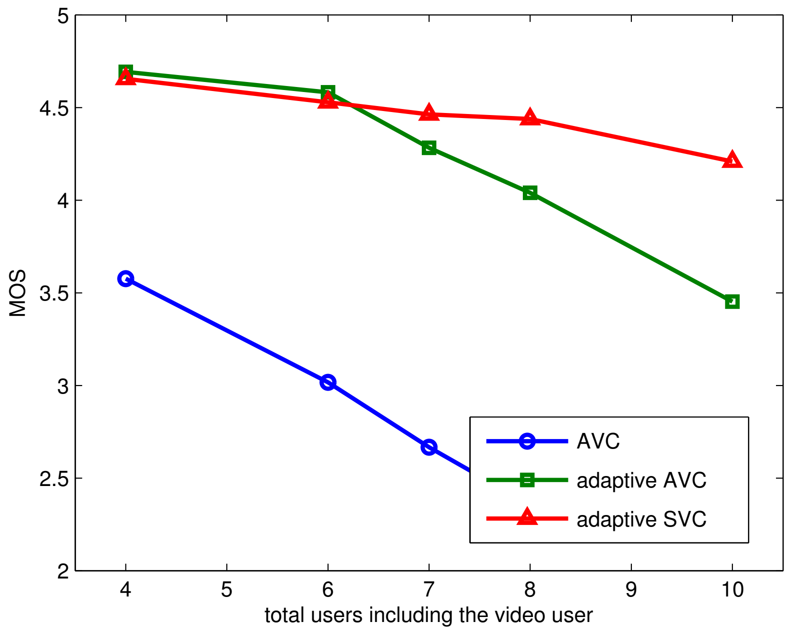


Figure 4.9: $MOS_{weighted}$ at different congestion levels for low-mobility scenario

the cell of all three cases are as shown in Figure 4.8. The non-adaptive AVC user suffers the most severe interruptions as expected, followed by that of the adaptive AVC and SVC respectively. The reason why the adaptive SVC user performs significantly better than the adaptive AVC case is because the SVC algorithm can afford to be very conservative when the channel becomes unfavorable by requesting only the small base-layer blocks well far in advance first. Once the channel/congestion conditions improve or that it has enough base-layer chunks stored in the window, it can then request for additional enhancement blocks to upgrade the existing chunks in the window to a higher layer later on. On the contrary, the adaptive AVC is not as flexible enough to use the same strategy since a higher representation of a chunk cannot reuse the existing lower representation. Specifically, the adaptive AVC user has to discard the entire existing lower representation once a better one is requested and received. Thus, it requires larger throughput to do so and runs a higher risk of playback interruptions compared to its SVC counterpart.

The estimated MOS plots, also referred to as $MOS_{weighted}$ in Section 2.2.3, for all three cases are shown in Figure 4.9. In the light load situation where the adaptive streaming user suffers almost no interruption, the $MOS_{weighted}$ for the adaptive AVC in such situation is slightly higher than that of the SVC due to better encoding efficiency of the AVC compared to the SVC. However, as the cell becomes more congested, e.g., there are more users in the cell and the streaming user starts to suffer more frequent and longer interruptions, the adaptive SVC algorithm is still able to maintain significantly higher $MOS_{weighted}$ than the adaptive AVC. From the operator's point of view, this translates into larger cell's capacity for video streaming services at the same targeted quality, or higher video quality at the

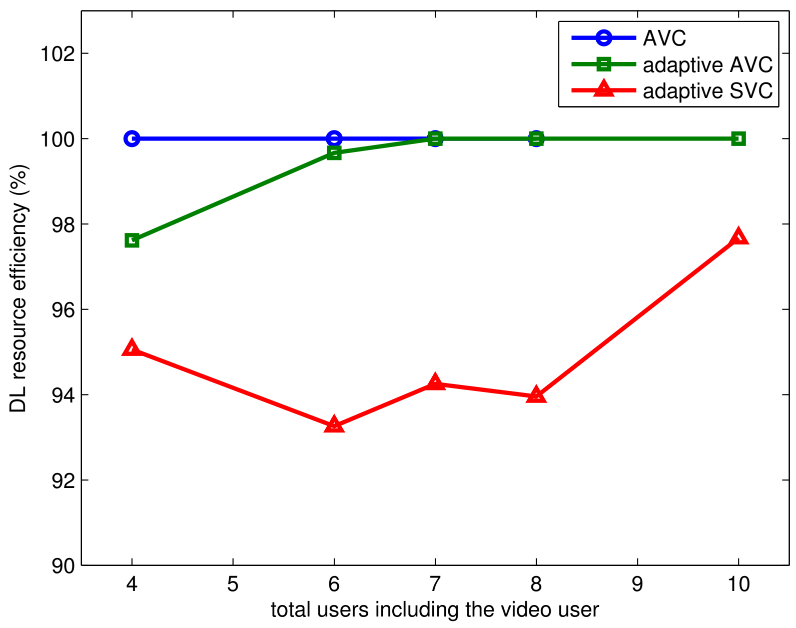


Figure 4.10: Downlink resource usage efficiency at different congestion levels for low-mobility scenario

same targeted amount of supported traffic load.

Finally, we investigate the downlink resource usage efficiency which is defined as

$$\text{DL resource efficiency (\%)} = 100 \cdot \frac{\text{number of decoded Bytes}}{\text{number of requested Bytes}}. \qquad (4.25)$$

Figure 4.10 depicts the efficiency plots for all three cases. The non-adaptive AVC is the most efficient streaming strategy since all the requested chunks are decoded. The adaptive AVC has a lower efficiency as the traffic load in the cell becomes less congested. This is because the better the channel/congestion situation is, the more likely the streaming user is able to fill all chunks in its adaptation window and come back to request for better representations to the existing chunks. The lower representations would then be discarded, thus lowering its resource usage efficiency. The adaptive SVC algorithm, however, appears to have the worst efficiency compared to the other two cases. This is due to the risk-taking behavior of the algorithm itself, e.g., the user might request for a block even though its success probability is less than 100% if the quality contribution from that particular block outweighs the risk of not getting it within the deadline. Nevertheless, the inefficiency in using the radio resources for both the adaptive AVC and SVC algorithms is minuscule, e.g., better than 90% over the entire simulation range with no significant discrepancy between both of them.

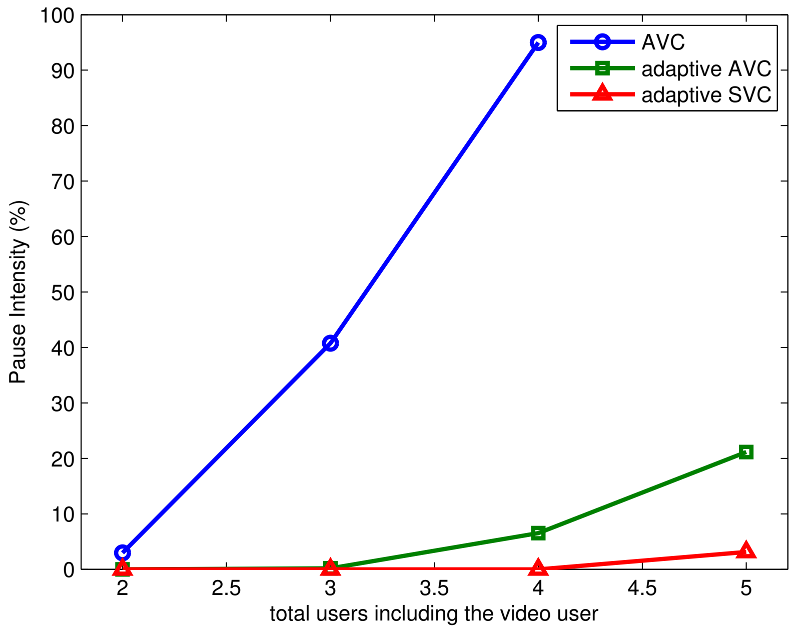


Figure 4.11: Pause intensity at different congestion levels for high-mobility scenario

4.5.3 High-Mobility Scenario

As a comparison with the previous section, the results from simulations with high-mobility users are discussed here. Figures 4.11, 4.12 and 4.13 show the Pause Intensity, the $MOS_{weighted}$ and the downlink resource usage efficiency of the streaming user at different cell congestion levels respectively. Similar to the results of the low-mobility scenario, the relative performances between all three streaming modes are the same. Specifically, the adaptive SVC yields the best performances in terms of minimizing interruptions and maintaining high $MOS_{weighted}$, followed by the adaptive AVC and non-adaptive AVC. For the resource usage efficiency plot, the ordering is in reverse as in the low-mobility scenario as well. The only obvious difference is, however, that the number of supported users in the cell has dropped significantly to attain the same performance compared to the low-mobility scenario. This is due to the lower cell throughput when users move at higher speed as expected.

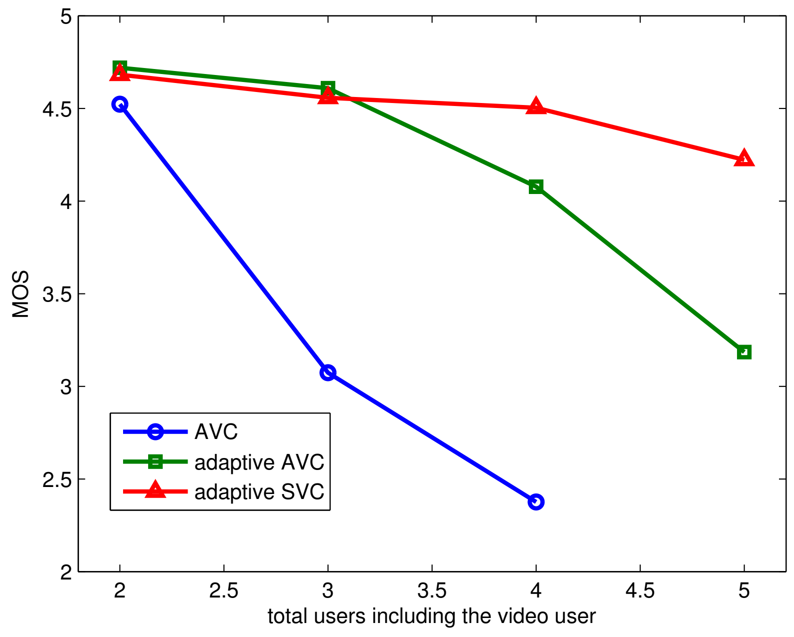


Figure 4.12: $MOS_{weighted}$ at different congestion levels for high-mobility scenario

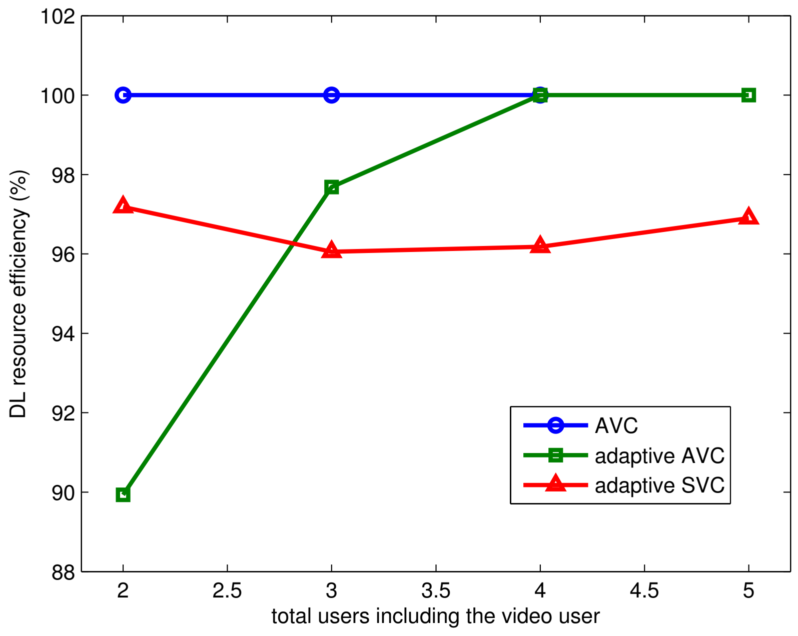


Figure 4.13: Downlink resource usage efficiency at different congestion levels for high-mobility scenario

Chapter 5

Conclusion and Outlook

Currently, billions of people around the world have access to high-speed mobile Internet via 3G technologies and beyond. With this number as well as the popularity of smart mobile devices expected to increase even further, mobile video streaming services will eventually contribute a large share of the overall traffic through a mobile network. However, providing good video streaming experience to users while keeping the congestion level in the network at a manageable level proves to be a challenging task due to the large bitrate and low delay requirements of such service. A large body of works have proposed several timestamp-based adaptive streaming architectures involving complex cross-layer communication/optimization across various network entities in the past. These solutions have seen so far, however, only limited success in real-world deployment. On the contrary, progressive download adaptive video streaming with HTTP/TCP which has re-emerged as an alternative concept recently is still relatively under explored, although it is expected to be a widely-accepted means to provide video services in the near future.

In this work, two adaptive streaming architectures, both for the timestamp-based and for the progressive downloading paradigms, designed for the dynamic and highly-fluctuating mobile environment have been proposed. They are also designed to work with and exploit the advantages of the Scalable Video Coding (SVC) extension to the H.264 video compression standard such that the video bitrate and quality can be scaled up or down with ease. In the following, major results for the proposed architectures for both streaming paradigms are summarized.

Timestamp-Based Adaptive Video Streaming

To provide timestamp-based adaptive video streaming with RTP/UDP, the Coordinated Adaptive Streaming (CAS) architecture has been proposed in Chapter 3. An additional network entity, referred to as the coordinated adaptation server, is required to be placed somewhere between the mobile network and the Internet. It is not a video streaming server by itself, but acts as a coordinated adaptation engine for all SVC-encoded video streams from the Internet toward the mobile streaming users within the same cell. The adaptation is done such that the videos can be jointly adapted periodically in a RD-optimized manner taking into account channel situations while being fair to other non-streaming users in the same time. For CAS to function properly, it is assumed that the RD information of the videos and generic information on the amount of radio resources used for each streaming user are made available to the coordinated adaptation server periodically. While the latter can also be considered as cross-layer information which the initial design concept tried to avoid, it is relatively much simpler to obtain than other proposals in the related works. Specifically, this information is obtainable at the mobile device itself where the streaming application, in theory, should be able to get such information from the underlying Physical layer with ease, provided that there is such an interface for this purpose between the two layers. Additionally, the amount of resources used can be in the form of radio resource chunks or timeslots allocated to the user in a certain period, thus is applicable to both OFDM-based and W-CDMA based technologies. Additionally, CAS does not assume to have complete knowledge of the resource usages for every users in the cell. It instead controls the congestion level of only the streaming traffic class to be within an acceptable level by adjusting the resource budget available to streaming users periodically.

Alternatively, a simplified adaptive architecture from CAS where a video for a streaming user is adapted independently from other users has also been proposed, referred to as the Uncoordinated Adaptive Streaming (UAS). The coordinated adaptation server is therefore no longer required for UAS, but the adaptation itself can be done by the originating server directly. UAS also requires neither the cross-layer information on the resource usage statistic nor the RD information of the video. The adaptation is performed based simply on the individual congestion indicator alone which is measurable entirely at the Application layer.

The performances of CAS and UAS in terms of the coordination gain, robustness to degrading channel conditions and their dependencies on other system parameters have been throughly discussed. The Iterative Efficient Set Approach (IEA) and the Steepest Ascend (SA) suboptimal search algorithms used to simplify the optimization problem in the CAS framework have been investigated in which it is found that the SA performs noticeably better than the IEA, albeit being much less complex. CAS also has been shown to be more tolerant toward adverse channel situations than UAS due to its ability to share radio resources within the group of streaming users. Additionally, the coordination gain has been demonstrated to increase with the number of streaming users and operating points in each video. It is also dependent on whether the resource scheduler at the base station is sensitive to the queuing delay at its transmission queues or not. The widely-used Proportional Fair scheduler which takes the packet delay as one of its criteria in allocating resources is therefore compatible and recommended to be used with CAS. Finally, simulation results have revealed that CAS and UAS are superior in terms of the resulting video quality than the current state of the art while the fairness to other non-streaming traffic is still preserved.

Progressive Download Adaptive Video Streaming

In Chapter 4, a progressive download adaptive streaming architecture for the mobile environment has been introduced. It has been designed to not only be compatible with the on-going Dynamic Adaptive Streaming over HTTP (DASH) standard, but also to exploit the various benefits of Scalable Video Coding and to address the challenges found in the mobile environment as well. To achieve such purpose, first a statistical model of the best-effort TCP throughput over a mobile channel has been constructed. It allows the client-based streaming application to estimate the success probability in getting a certain number of Bytes through the current mobile channel within a limited time by observing statistical properties of the channel in the recent past. With the ability to make these estimations for different parts of the video, referred to as blocks in this work, the streaming application can make a smart decision which missing blocks to request from the server and in which order by weighting the benefits different blocks contribute to the overall video quality and the risks in obtaining them. To further reduce the complexity, an iterative sub-optimum algorithm to determine the best request sequence has also been introduced. The initial request sequence to start the search can also be chosen based on a rough estimate of the current congestion level as well to help improving the performance of the search algorithm.

Simulation results between the proposed architecture and the current state-of-the-art progressive download adaptive streaming based on having multiple non-scalable AVC representations have shown significant improvements both in terms of the reduction in interruptions and the perceived video streaming quality by the users. This is also true in a high-mobility scenario where the streaming user moves at a vehicular speed of 30 km/h.

Outlook

During the course of this work, several interesting related research areas have been identified that could be of great contributions to the overall body of knowledge and developments of future adaptive video streaming architectures. In the following, these potential topics suitable for further research as well as the limitations of the proposed adaptive architectures are discussed.

In Section 2.2.3, several Key Performance Indicators (KPI) used in this work have been introduced. One of them is the estimated Mean Opinion Score (MOS), referred to as $MOS_{weighted}$, which is the original MOS of the video weighted by the effects of the interruptions during the playback in the form of a multiplicative scaling factor. However, the used degradation model has been derived from a limited set of subjective test results which were conducted using low-quality low-bitrate videos. This tends to hide the true extend of how severe playback interruptions can degrade the perceived video quality as the original low-bitrate contents already have bad perceived quality by the users to begin with. Although it has been suggested by several other works that this degradation effects would be more severe with high-quality contents, similar to those used in this work, there has been no comprehensive study on this subject to date yet. The $MOS_{weighted}$ in this work therefore serves only as an upper bound on the true MOS, e.g., as an optimistic approximation of the user's satisfaction toward the streaming service. Having better understanding of the negative effects that the interruptions, as well as the frequency, location and duration of such events have on the MOS for a broad range of video content is therefore of great interest.

In the CAS framework, the search algorithms to determine the $\vec{C}_{opt}$ by the coordinated adaptation server investigated in Section 3.3.3 are guaranteed to converge to the true global optimum only with the continuous and non-convex assumptions of the RD curves. Although the RD curves of most videos exhibit the non-convex characteristic and that the Steepest Ascend (SA) variation of the Greedy algorithm has been shown to perform well with discrete RD curves, having a low-complexity search algorithm with good performance with any type of the RD curve would be beneficial. This is especially true if the SVC-encoded videos contain more than one spatial layer as the RD characteristics of such contents often show signs of convexity. The Polyblock algorithm which is guaranteed to find the true global optimum even with discrete and convex RD curves still converges too slowly and is much too complex. However, it is still interesting to investigate if this can be used as a starting point for further modifications or not. The CAS also relies on having an interface for the mobile terminal's Application layer to its Physical layer to obtain radio resource usage information. Although this interface is theoretically easy to be implemented and should be useful for other purposes and/or other adaptive streaming architectures as well, it still relies on the hardware manufacturers to provide such a feature. In addition, the dependency of using a delay-sensitive resource scheduler such as the Proportional Fair at the base station for the CAS to be effective in influencing resource allocations to different users is another restriction of this architecture.

Some interesting improvements that are worth further investigation for the proposed progressive download adaptive streaming architecture in this work are as follows. First, having parallel HTTP requests to download multiple blocks simultaneously is an interesting possibility as the combined throughput from multiple HTTP/TCP connections is likely to be less fluctuated. Sudden throughput reduction from a single connection due to, e.g., the TCP's loss recovery mechanism, will not affect other parallel connections. To accommodate this technique, the algorithm to determine the best request sequence $\widehat{S}_{opt}$ as well as the concept of the request sequence itself where video blocks are requested sequentially must be modified. Second, the statistical model of the TCP throughput over the mobile channel has been developed from analyzing various TCP throughput traces of a low-mobility user moving at 3 km/h. It is interesting whether modifications are necessary or if the more fundamental changes must be made to the statistical model so that it is applicable with different user's speed or not. Finally, the proposed architecture only performs the adaptation individually based on the user's own channel situation and RD information. Coordinated adaptation amongst different streaming users in the cell is currently not possible with the current algorithm, although it should be theoretically feasible, e.g., by means of having a "price" for each radio resource unit, adjusted by a centralized entity in relation to the overall congestion situation.

List of Figures

List of Tables

Bibliography

[1] ABI Research, "Two Billion Covered by 3G and 4G Data Services," [Online]. Available: http://www.abiresearch.com/press/3562-Two+Billion+Covered+by+3G+and+4G+Data+Services.

[2] Cisco Systems, Inc., "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2010-2015," [Online]. Available: http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.html.

[3] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard," *IEEE Transaction on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1103–1120, September 2007.

[4] Apple, Inc., "Darwin Streaming Server," [Online]. Available: http://dss.macosforge.org/.

[5] RealNetworks, "Helix Universal Media Server," [Online]. Available: http://www.realnetworks.com/helix/helix-universal-streaming-media-server.aspx.

[6] Akamai Technologies, Inc., "Apple and Akamai Create High Quality Network for Internet Streaming," [Online]. Available: http://www.akamai.com/html/about/press/releases/1999/press_072199.html.

[7] Adobe Systems, Inc., "Adobe Flash Player," [Online]. Available: http://www.adobe.com/products/flashplayer/.

[8] Web Hypertext Application Technology Working Group and World Wide Web Consortium, "HTML5 Video," [Online]. Available: http://html5video.org/.

[9] ISO/IEC JTC 1/SC 29/WG 11 (MPEG), "Dynamic Adaptive Streaming over HTTP," Guangzhou, China, October 2010.

[10] K. Tappayuthpijarn, G. Liebl, T. Stockhammer, and E. Steinbach, "Adaptive Video Streaming over a Mobile Network with TCP-Friendly Rate Control," in *Proceedings of the International Wireless Communications and Mobile Computing Conference*, June 2009.

[11] K. Tappayuthpijarn, T. Stockhammer, and E. Steinbach, "A Novel Coordinated Adaptive Video Streaming Framework for Scalable Video over Mobile Networks," in *Proceedings of the IEEE International Conference on Image Processing*, September 2010.

[12] K. Tappayuthpijarn, T. Stockhammer, and E. Steinbach, "HTTP-Based Scalable Video Streaming over Mobile Networks," in *Proceedings of the IEEE International Conference on Image Processing*, September 2011.

[13] C. Smith and D. Collins, *3G Wireless Networks*. McGraw-Hill Professional, September 2001.

[14] 3GPP, "Universal Terrestrial Radio Access Network (UTRAN) - Overall Description Release '99," 3rd Generation Partnership Project (3GPP), TS 25.401, v3.10.0, 2002.

[15] 3GPP, "Universal Terrestrial Radio Access Network (UTRAN) - Overall Description Release 5," 3rd Generation Partnership Project (3GPP), TS 25.401, v5.10.0, 2005.

[16] 3GPP, "Universal Terrestrial Radio Access Network (UTRAN) - Overall Description Release 7," 3rd Generation Partnership Project (3GPP), TS 25.401, v7.6.0, 2008.

[17] 3GPP, "Evolved Universal Terrestrial Radio Access Network (E-UTRAN) - Overall Description Release 8," 3rd Generation Partnership Project (3GPP), TS 36.300, v8.12.0, 2010.

[18] International Telecommunication Union - Radio Communication Sector, "ITU global standard for international mobile telecommunications ´IMT-Advanced´," [Online]. Available: http://www.itu.int/ITU-R/index.asp?category=information&rlink=imt-advanced&lang=en.

[19] 3GPP, "Evolved Universal Terrestrial Radio Access Network (E-UTRAN) - Overall Description Release 10," 3rd Generation Partnership Project (3GPP), TS 36.300, v10.4.0, 2011.

[20] IEEE, "Air Interface for Fixed and Mobile Broadband Wireless Access Systems; Amendment for Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands," Institute of Electrical and Electronics Engineers (IEEE), IEEE 802.16e, 2005.

[21] V. Vukadinovic and G. Kalrsson, "Video Streaming Performance under Proportional Fair Scheduling," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 3, April 2010.

[22] Nomor Research GmbH, "NeatBox : Network Emulator for Application Testing," [Online]. Available: http://nomor.de/home/solutions-and-products/products.

[23] G. J. Sullivan and T. Wiegand, "Video Compression - From Concepts to H.264/AVC Standard," in *Proceedings of the IEEE*, January 2005.

[24] International Telecommunication Union - Telecommunication Standardization Sector, "H.264 - Series H: Audio Visual and Multimedia Systems, Advanced Video Coding for Generic Audiovisual Services," March 2010, [Online]. Available: http://www.itu.int/rec/T-REC-H.264/en.

[25] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC Video Coding Standard," *IEEE Transaction on Circuits and Systems for Video Technology*, vol. 13, no. 6, pp. 560–576, July 2003.

[26] G. Liebl, K. Tappayuthpijarn, T. de Moraes, K. Grueneberg, C. Hellge, T. Schierl, C. Keip, H. Stadali, and N. Pham, "Study of the Performance of Scalable Video Coding over a DVB-SH Satellite Link," in *Proceedings of the IEEE International Conference on Multimedia and Expo*, July 2010.

[27] G. Liebl, K. Tappayuthpijarn, T. Schierl, K. Grueneberg, H. Stadali, C. Keip, and N. Pham, "Statistical Multiplexing for SVC over DVB-SH," in *Proceedings of the 10th Workshop Digital Broadcasting*, September 2009.

[28] I. Amonou, N. Cammas, S. Kervadec, and S. Pateux, "Optimized Rate-Distortion Extraction with Quality Layers in the Scalable Extension of H.264/AVC," *IEEE Transaction on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1186–1193, September 2007.

[29] S. Winkler and P. Mohandas, "The Evolution of Video Quality Measurement: From PSNR to Hybrid Metrics," *IEEE Transaction on Broadcasting*, vol. 54, pp. 660–668, September 2008.

[30] T. Porter and X. H. Peng, "An Objective Approach to Measuring Video Playback Quality in Lossy Networks using TCP," *IEEE Communications Letters*, vol. 15, pp. 76–78, September 2010.

[31] M. H. Pinson and S. Wolf, "A New Standardized Method for Objectively Measuring Video Quality," *IEEE Transaction on Broadcasting*, vol. 50, pp. 312–322, September 2004.

[32] S. Wolf and M. Pinson, "Video Quality Measurement Techniques," June 2002, [Online]. Available: http://www.its.bldrdoc.gov/n3/video/documents.htm.

[33] A. A. Webster, C. T. Jones, and M. H. Pinson, "An Objective Video Quality Assessment System Based on Human Perception," in *Proceedings of the Society of Photographic Instrumentation Engineers*, September 1993.

[34] "Video Quality Metric (VQM) Software," [Online]. Available: http://www.its.bldrdoc.gov/vqm/.

[35] X. Tan, J. Gustafsson, and G. Heikkilae, "Perceived Video Streaming Quality under Initial Buffering and Rebuffering Degradations," in *Proceedings of MESAQIN (Mea-*

surement of Speech, Audio and Video Transmission Quality In Telecommunication Networks), June 2006.

[36] J. Gustafsson, G. Heikkiläm, and M. Pettersson, “Measuring Multimedia Quality in Mobile Networks with an Objective Parametric Model,” in *Proceedings of the IEEE International Conference on Image Processing*, October 2008.

[37] R. Mok, E. Chan, and R. Chang, “Measuring the Quality of Experience of HTTP Video Streaming,” in *Proceedings of the IEEE/IFIP International Symposium on Integrated Network Management*, May 2011.

[38] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image Quality Assessment: From Error Visibility to Structural Similarity,” *IEEE Transaction on Image Processing*, vol. 13, pp. 600–612, April 2004.

[39] Z. Wang and A. C. Bovik, “A Human Visual System-Based Objective Video Distortion Measurement System,” in *Proceedings of the International Conference on Multimedia Processing and System*, 2001.

[40] A. Begen, T. Akgul, and M. Baugher, “Watching Video over the Web,” *IEEE Internet Computing*, vol. 15, March 2011.

[41] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, “RTP: A Transport Protocol for Real-Time Applications,” Internet Engineering Task Force, RFC 3550, July 2003.

[42] M. Handley, V. Jacobson, and C. Perkins, “SDP: Session Description Protocol,” Internet Engineering Task Force, RFC 4566, July 2006.

[43] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, A. Peterson, R. Sparks, M. Handley, and E. Schooler, “SIP: Session Initiation Protocol,” Internet Engineering Task Force, RFC 3261, June 2002.

[44] H. Schulzrinne, A. Rao, and R. Lanphier, “Real Time Streaming Protocol (RTSP),” Internet Engineering Task Force, RFC 2326, April 1998.

[45] J. Y. Le Boudec and P. Thiran, *Network Calculus: A Theory for Deterministic Queuing Systems for the Internet.* Springer, 2001.

[46] Q. Zhang, W. Zhu, and Y. Q. Zhang, “End-to-End QoS for Video Delivery over Wireless Internet,” vol. 93, January 2005, pp. 123–134.

[47] P. A. Chou and Z. Miao, “Rate-Distortion Optimized Streaming of Packetized Media,” *IEEE Transaction on Multimedia*, vol. 8, pp. 390–404, April 2006.

[48] S. Thakolsri, W. Kellerer, and E. Steinbach, “QoE-Based Cross-Layer Optimization of Wireless Video with Unperceivable Temporal Video Quality Fluctuation,” in *Proceedings of the IEEE International Conference on Communications*, June 2011.

[49] S. Thakolsri, S. Khan, E. Steinbach, and W. Kellerer, "QoE-Driven Cross-Layer Optimization for High Speed Downlink Packet Access," *Journal of Communications*, vol. 4, pp. 669–680, October 2009.

[50] S. Thakolsri, W. Kellerer, and E. Steinbach, "QoE-Based Rate Adaptation Scheme Selection for Resource-Constrained Wireless Video Transmission," in *Proceedings of the ACM International Conference on Multimedia*, October 2010.

[51] G. Liebl, W. Tu, and E. Steinbach, "Proxy-Based Transmission Strategies for Wireless Video Streaming," in *Proceedings of the IEEE Packet Video*, December 2007.

[52] J. Huang, Z. Li, M. Chiang, and A. K. Katsaggelos, "Joint Source Adaptation and Resource Allocation for Multi-User Wireless Video Streaming," *IEEE Transaction on Circuits and Systems for Video Technology*, vol. 18, pp. 582–595, May 2008.

[53] M. Chiang, S. Low, A. Calderbank, and J. Doyle, "Layering as Optimization Decomposition: A Mathematical Theory of Network Architectures," *Proc. IEEE*, 2007.

[54] D. Palomar and M. Chiang, "A Tutorial on Decomposition Methods for Network Utility Maximization," *IEEE Journal on Selected Areas in Communications*, August 2006.

[55] Z. Li, J. Huang, and A. Katsaggelos, "Pricing-Based Collaborative Multi-User Video Streaming Over Power Constrained Wireless Downlink," in *Proc. IEEE ICASSP*, May 2006.

[56] J. Huang, Z. Li, and M. Chiang, "Pricing-Based Rate Control and Joint Packet Scheduling for Multi-User Wireless Uplink Video Streaming," in *Proc. IEEE PV*, April 2006.

[57] E. Kohler, M. Handley, and S. Floyd, "Datagram Congestion Control Protocol (DCCP)," Internet Engineering Task Force, RFC 4340, March 2006.

[58] M. Handley, S. Floyd, J. Padhye, and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification," Internet Engineering Task Force, RFC 3448, January 2003.

[59] S. Floyd, E. Kohler, and J. Padhye, "Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 3: TCP-Friendly Rate Control (TFRC)," Internet Engineering Task Force, RFC 4342, March 2006.

[60] N. Laoutaris and I. Stavrakakis, "Intrastream Synchronization for Continuous Media Streams: A Survey of Playout Schedulers," *IEEE Network*, vol. 16, pp. 30–40, June 2002.

[61] T. H. Luan, L. X. Cai, and X. Shen, "Impact of Network Dynamics on User's Video Quality: Analytical Framework and QoS Provision," *IEEE Transaction on Multimedia*, vol. 12, pp. 64–78, January 2010.

[62] Y. Li, A. Markopoulou, J. Apostolopoulos, and N. Bambos, "Content-Aware Playout and Packet Scheduling for Video Streaming over Wireless Links," *IEEE Transaction on Multimedia*, vol. 93, pp. 885–895, August 2008.

[63] J. Park, H. Lee, and S. Lee, "Cross-Layer Optimization for Scalable Video Coding and Transmission over Broadband Wireless Networks," in *Proceedings of the IEEE International Conference on Image Processing*, September 2007.

[64] G. Carneiro, J. Ruela, and M. Ricardo, "Cross-Layer Design in 4G Wireless Terminals," *IEEE Transaction on Wireless Communications*, pp. 7–13, April 2004.

[65] T. Yoshimura, T. Ohya, T. Kawahara, and M. Etoh, "Rate and Robustness Control with RTP Monitoring Agent for Mobile Multimedia Streaming," in *Proceedings of the IEEE International Conference on Communications*, August 2002.

[66] A. Vetro and C. Timmerer, "Digital Item Adaptation: Overview of Standardization and Research Activities," *IEEE Transaction on Multimedia*, vol. 7, pp. 418–426, June 2005.

[67] S. Devillers, C. Timmerer, J. Heuer, and H. Hellwagner, "Bitstream Syntax Description-Based Adaptation in Streaming and Constrained Environments," *IEEE Transaction on Multimedia*, vol. 7, pp. 463–470, June 2005.

[68] D. Mukherjee, E. Delfosse, J. G. Kim, and Y. Wang, "Optimal Adaptation Decision-Taking for Terminal and Network Quality of Service," *IEEE Transaction on Multimedia*, vol. 7, pp. 454–462, June 2005.

[69] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol - HTTP/1.1," Internet Engineering Task Force, RFC 2616, June 1999.

[70] C. Albanesius, "Netflix Working on HTML5 Standard for Streaming Video," [Online]. Available: http://www.pcmag.com/article2/0,2817,2374694,00.asp.

[71] S. Akhshabi, A. C. Begen, and C. Dovrolis, "An Experimental Evaluation of Rate-Adaptation Algorithms in Adaptive Streaming over HTTP," in *Proceedings of the ACM Multimedia Systems*, February 2011.

[72] J. Yao, S. Kanhere, I. Hossain, and M. Hassan, "Empirical Evaluation of HTTP Adaptive Streaming under Vehicular Mobility," in *Proceedings of the IFIP International Conferences on Networking*, vol. 1, 2011.

[73] Y. Sánchez, T. Schierl, C. Hellge, T. Wiegand *et al.*, "iDASH: improved Dynamic Adaptive Streaming over HTTP using Scalable Video Coding," in *Proceedings of the ACM Multimedia Systems*, February 2011.

[74] C. Liu, I. Bouazizi, and M. Gabbouj, "Rate Adaptation for Adaptive HTTP Streaming," in *Proceedings of the ACM Multimedia Systems*, February 2011.

[75] C. Liu, I. Bouazizi, and M. Gabbouj, "Parallel Adaptive HTTP Media Streaming," in *Proceedings of the IEEE International Conference on Computer Communications and Networks*, August 2011.

[76] R. Pantos, "HTTP Live Streaming," Internet Engineering Task Force, Internet Draft, September 2011.

[77] V. Ramshankar, "QoS Support for Dynamic Adaptive Streaming over Mobile HTTP," Ph.D. dissertation, Technische Universitaet Muenchen, 2011.

[78] F. Hartanto, J. Kangasharju *et al.*, "Caching Video Objects: Layers vs Versions?" in *Proceedings of the IEEE International Conference on Multimedia and Expo*, August 2002.

[79] P. Cuetos *et al.*, "Adaptive Rate Control for Streaming Stored Fine-Grained Scalable Video," in *Proceedings of the International Workshop on Network and Operating Systems Support for Digital Audio and Video*, May 2002.

[80] P. Cuetos, P. Guillotel, and K. W. Ross, "Implementation of Adaptive Streaming of Stored MPEG-4 FGS Video over TCP," in *Proceedings of the IEEE International Conference on Multimedia and Expo*, August 2002.

[81] R. Kuschnig, I. Kofler, and H. Hellwagner, "An Evaluation of TCP-Based Rate-Control Algorithms for Adaptive Internet Streaming of H.264/SVC," in *Proceedings of the ACM Multimedia Systems*, February 2010.

[82] L. Choi, M. Ivrlac, E. Steinbach, and J. Nossek, "Sequence-Level Models for Distortion-Rate Behaviour of Compressed Video," in *Proceedings of the IEEE International Conference on Image Processing*, November 2005.

[83] J. Brehmer and W. Utschick, "A Decomposition of the Downlink Utility Maximization Problem," in *Proceedings of the IEEE Annual Conference on Information Sciences and Systems*, November 2005.

[84] J. Brehmer and W. Utschick, "Nonconcave Utility Maximization in the MIMO Broadcast Channel," *EURASIP Journal on Advances in Signal Processing*, 2009.

[85] H. Tuy, M. Minoux, and N. Hoai-Phuong, "Discrete Monotonic Optimization with Application to a Discrete Location Problem," *SIAM Journal on Optimization*, vol. 17, 2006.

[86] W. Kaplan, *Maxima and Minima with Applications: Practical Optimization and Duality.* Wiley, November 1998.

[87] Y. Tian, K. Xu, and N. Ansari, "TCP in Wireless Environments: Problems and Solutions," *IEEE Communications Magazine*, vol. 43, pp. 32–47, March 2005.

[88] L. F. Fenton, "The Sum of Lognormal Probability Distributions in Scatter Transmission Systems," *IRE Transactions on Communication Systems*, vol. 8, pp. 57–67, 1960.