SCALABLE VIDEO TRANSPORTATION

USING LOOK AHEAD SCHEDULING

A thesis submitted for the degree of Master of Philosophy

School of Computer Science and Electronic Engineering

University of Essex

December 2017

Abstract

This thesis introduces many video streaming algorithms and techniques to improve the performance of scalable video dissemination over a packet network. As the quality of visual content in a best-effort network is susceptible to average bandwidth availability, packet loss, packet delay and packet delay variation (jitter), the study proposes a scheduling algorithm and buffering technique.

To combat the variation of delay (jitter) in a best-effort network, the study proposes Look Ahead Scheduling Algorithm (LASA) which provides unequal look-ahead by safeguarding the base layer of the scalable video. The results of the experiment show that LASA reduces the number of frames with violate deadlines and significantly improves the continuity of video stream without compromising the average Y-PSNR. The LASA is also extended to support muti-server scalable video streaming. The performance evaluation is investigated by using a network simulation tool.

This thesis also presents the weighted multi-playback buffer management system for scalable video streaming in a congested network. The approach intrinsically provides weighted protection to the more important video layers within a video stream. The study compares the performance of a weighted scheme with a multi-playback buffer which gives equal protection to every video layer. The simulation results show that weighted multi-playback buffer management can improve the average Y-PSNR and the continuity of the scalable video stream.

Acknowledgements

I would like to express my gratitude to all people who supported me throughout the period of my study at the University of Essex, UK. I am extremely grateful to both my supervisors: Dr John Woods, and Professor Mohammed Ghanbari for their guidance, helpful comments on my work, and wise advice throughout the entirety of research. I would also like to thank all members of my supervisory board, who encouraged and gave me the potential advice and support. I sincerely thank all of my reviewers for their positive reviews of my publications.

Finally, I would like to thank all staff at the University of Essex for all of your help and support. Although their names do not appear here, I would like to let them all know that I owe them a great debt of gratitude. Thank you all for your support.

Table of Contents

Page

List of Figures

Page

List of Tables

Page

List of Abbreviations

CBR	Constant Bit Rate
VBR	Variable Bit Rate
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
RTP	Real-time Protocol
HTTP	Hypertext Transfer Protocol
PDV	Packet Delay Variation
DASH	Dynamic Adaptive Streaming over HTTP
NAL	Network Abstraction Layer
ADSL	Asymmetric digital subscriber link
ACK	acknowledgement
B-Frame	Bidirectional Frame
P-Frame	Predictive Frame
I-Frame	Intra-Frame
FIFO	First-in, First-out
FTP	File Transfer Protocol
FEC	Forward Error Correction
ITU-T	International Telecommunication Union-Telecommunication Standardization Sector
MPEG	Motion Picture Experts Group
SNR	Signal-to-Noise Ration
PSNR	Peak Signal-to-Noise Ration
QoE	Quality of Experience

Chapter 1
Introduction

1.1 Thesis Motivation

Video delivery is one of the most active research areas in the field of wired and wireless networks. Transferring a stream of video needs special requirements both in the field of the computer networks and video coding to achieve the continuity of the video stream. The challenge in this research is how to maintain the continuity of video along with the quality of video in the jitter-prone network.

The original idea of scalable video coding was proposed by [1] in late 1980 to cope with the problem. Scalable video would be composed of one base layer and many enhancement layers. Forwarding base layer packets need minimum link capacity, but if there is more bandwidth available, enhancement layer packets will be carried to increase video quality appearing to the end users. For this reason, the scalable video can tolerate the bandwidth variation [2].

In the last decade, a number of research in video scheduling has been carried out, but very little effort has been dedicated to examining the low-complexity scheduling algorithm. This thesis introduces low complexity scalable video scheduling algorithms and techniques to improve the performance of scalable video delivery over packet networks. As the quality of visual content in a best-effort network is susceptible to average bandwidth availability, packet loss, packet delay and packet delay variation (jitter), the study proposes scheduling algorithm and the buffering technique.

To combat the variation of delay (jitter) in the best-effort network, this thesis proposes the Look Ahead Scheduling Algorithm (LASA) which provides unequal look-ahead by safeguarding the base layer of the scalable video. The aim of this thesis is to reduce the number of frames which

violate deadlines that improve the continuity of video stream without compromising the average Y-PSNR. The LASA is also extended to support multiple server scalable video streaming. The performance evaluation is investigated by using a network simulation tool.

This thesis also presents the weighted multi-playback buffer management system for scalable video streaming in a congested network. The approach intrinsically provides weighted protection to the more important video layers within a video stream. The study compares the performance of a weighted scheme with a multi-playback buffer that gives equal protection to every video layer. The simulation results show that weighted multi-playback buffer management can improve the average Y-PSNR and the continuity of the scalable video stream.

1.2 Thesis Contribution and Organization

This thesis is organized as follow:

Chapter 2 presents background literature relating to video streaming. In addition, the overview of buffer management and scalable video is briefly mentioned. Furthermore, the basic idea of the scheduling algorithm is introduced.

Chapter 3 proposes weighted multi-playback buffer management for scalable video streaming that provides multilevel protection to playout buffers. The proposed buffer management defines the new paradigm to buffer management. Rather than handling the buffer as one area of memory, the proposed algorithm splits the buffer and allocates to video layers.

Chapter 4 proposes Look Ahead Scheduling Algorithm (LASA) that transforms the shape of the sending buffer from a rectangular sliding window to a trapezoid shape sending window. The look ahead scheduling algorithm provides multiple levels of protection to each video layer.

Chapter 5 proposes the adaptive look ahead scheduling algorithm (ALASA)-the adaptive version of LASA. The ALASA is able to adapt its sending buffer shape to suit the bandwidth fluctuation.

The results show that the ALASA can resist the sharp decrease in bandwidth availability so the ALASA can reduce the percentage of freeze frames and maintain the quality of video.

Chapter 6 proposes the ALASA in multiple server streaming. This algorithm can avoid the duplication in the scheduling process and can adapt the sending buffer to match the network condition.

Chapter 7 proposes the dropping policy that makes a decision whether the video portion should be sent or should be dropped. By concerning dependency relation between the video frames, our proposed dropping policy can drop some video data when network conditions change but still maintain the continuity of video.

Finally, *Chapter 8* presents the conclusion and Future Direction.

1.3 List of Publications

Journals

- A. Palawan, J. Woods, and M. Ghanbari, "Continuity-Aware Scheduling Algorithm for Scalable Video Streaming," Computers, vol. 5 (special Issue "Selected Papers from the 7th CEEC), p. 11, 2016. (Published)

- A. Palawan, J. Woods, and M. Ghanbari, "Continuity-Aware Scalable Video Streaming Using an Adaptive Look Ahead Scheduling Algorithm," Signal Processing: Image Communication (under review).

Conferences

- A. Palawan, J. Woods, and M. Ghanbari, "Weighted multi-playback buffer management for scalable video streaming," in Computer Science and Electronic Engineering Conference, 2014. (Published)

- A. Palawan, J. Woods, and M. Ghanbari, "A Jitter-Tolerant Scheduling Algorithm to Improve Continuity in Scalable Video Streaming," in Computer Science and Electronic Engineering Conference, 2015. (Published)

Chapter 2
Background Literature

2.1 Challenges in Video Delivery

This section explains the challenges in video delivery. The approaches to video delivery will be briefly described. In section 2.1.1, we will explain the traditional video delivery of downloading the whole video file and discuss its advantage and disadvantage. Next, section 2.2.2 describes the overview of video streaming where we will discuss the delivery technique and relating protocols.

2.1.1 Video Delivery by Downloading the Whole Video

The straightforward type of video delivery is downloading a whole file before playback. This approach employs the reliable transport layer protocol such as TCP and application protocol such as FTP or HTTP. The compressed video file will be downloaded in the same manner as other files. Figure 2.1 a) demonstrates how the compressed video is downloaded using a Web server. The browser sends GET command and waits for the Response message. After the video file is finished downloading, it is transferred to the video player to decode and display.

The drawback of this approach is waiting time (as shown in Figrue 2.1 b) which is the time before displaying the first frame of the video because the browser needs to finish downloading the whole file before displaying [3-5]. Note that a whole movie might need an hour before displaying the first frame depending on the file size and bandwidth avalability. Video delivery via downloading the whole file cannot satisfy users in terms of startup delay time, which is one of the

top three challenges in delivering videos. These challenges are firstly, how to reduce startup delay time (waiting time), secondly, how to minimize the possibility of stall event, thirdly, how to mazimize the video quality. Moreover, the user cannot use forward, backward and skip on the video until the video file is download completely. Another drawback is waste of bandwidth. The waste of bandwidth noticeably increases if the user has to download the whole file, but the content of the video cannot make a good impression on the user who then stops watching this video.

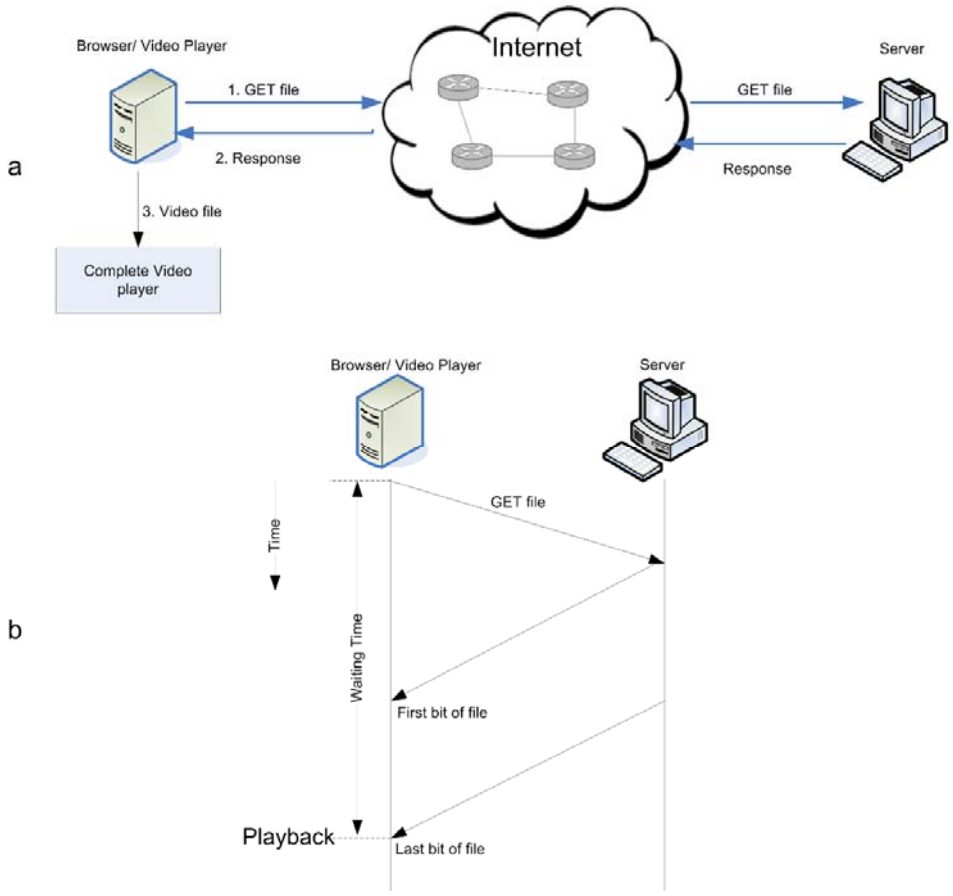

Figure 2.1 Downloading Whole Video Mechanism

a) Application layer mechanism to download a file.

b) Waiting time before starting playback.

However, downloading the whole video before playing can guarantee that there is no stall event and the video quality is equal to the encoded video at the server.

In the next section, we will consider another approach of delivering video - video streaming.

2.1.2 Video Streaming

Another approach to video delivery is using streaming, where the user can play videos while sending and receiving process are continuing. Therefore, the user does not need to wait until the downloading process finishes. The fundamental concept of video streaming is to download the parts of the video and once the initial receiver's buffer is filled, the video will be played[6]. This method allows the receiver to decode the video parts received and display the decoded picture while other parts of the video are arriving. We can briefly explain that the video streaming comprises of four main steps. Firstly, to separate the compressed video into parts and packetize them before transmission, secondly, begin to deliver them to the receiver, thirdly, the packets will be decoded if these packets arrive at the receiver correctly and no later than their decoding deadline, and finally, the decoded picture will be displayed.

However, in contrast to the downloading approach, video streams need additional modules to achieve their goal. Figure 2.2 shows the overview architecture of video streaming. To achieve video delivery while displaying the video, many essential modules may be required. For example, the receive buffer is a vital part of video streaming because bandwidth fluctuation in a shared network such as the Internet may lead to the variable incoming rate at the receiver. Thus, video streaming needs a buffer to temporarily store the video packets before playout [7] [8]. Another important module is called "send buffer" where the encoded video packets will be filled in. The

scheduler handles the video data in the send buffer by selecting the most suitable video packet to transmit or dropping some video packets [9-11]. Furthermore, the feedback messages which report channel and receiver conditions such as round trip time (RTT), packet error rate (PER), throughput, and buffer level status, will be taken into consideration to adjust parameters of scheduling strategy, packet selection strategy, and resource allocation strategy.

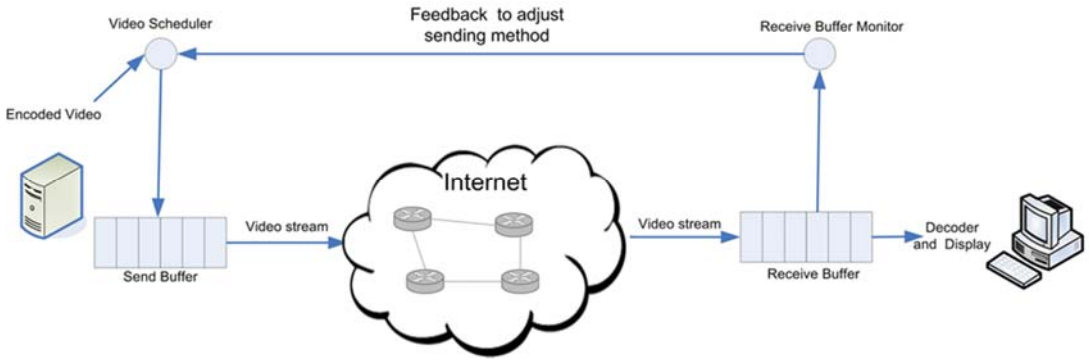

Figure 2.2 Overview Architecture of Video Streaming.

The bitrate adaptability to suit the bandwidth availability is a basic requirement of the video streaming system. The three main techniques that support bitrate adaptation are stream switching or multiple bit rates (MBR), transcoding, and scalable video coding [12].

Firstly, stream switching technique, or multiple bit rates (MBR), is the process where the raw input video is encoded into multiple encoded streams.The output of this process is a set of video files with different bitrate. To stream the video, the scheduler will select an encoded stream that suits the bandwidth availability the most and might switch to another encoded stream if bandwidth availability has changed. This technique needs the least processing complexity on the streaming server because when the encoded streams are produced, there are no further processing requirements relating to the encoded streams. The disadvantage of this technique is that more storage is needed to store all versions of the encoded streams.

The second technique is transcoding. It is the technique in adapting the bit rate of videos on the fly to match the currently available bandwidth. Transcoding algorithms adjusts the parameters of the compression algorithm such as video resolution, frame rate or video quality that causes the resulting bit rate of the video to be varied. The main drawback of this technique is that it needs high processing power.

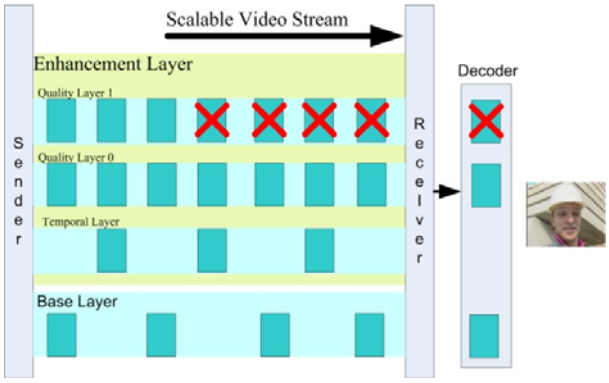

Figure 2.3 An example of possible NAL units removed from the scalable video stream.

The third technique is scalable video coding. This coding technique produces a multi-layer bit stream, one carrying the most critical video data which is called the base layer. The others, named enhancement layer, carry the extended data of the base layer. The encoded stream can be adapted on the fly. Figure 2.3 demonstrates the concept of scalable video coding where some units in the video stream can be dropped on the fly to match current available bandwidth.

The adaptable video stream has been mentioned already. Next, we will discuss the techniques and protocols to transmit the video stream. Firstly, there is a method of video streaming called “progressive download” that employs HTTP/TCP to download video files and display them simultaneously [13]. In the progressive download, the video player start displaying after the initial playout buffer is filled. Thus, initial waiting time is reduced significantly compared to the case of downloading the whole video. The startup time of displaying a video has a negative effect on user

satisfaction [14]. In [15] reported that every 1-second increase in startup time results in approximately 6% of abandonment rate.

In practice, the encoded video in adaptive video streaming will be encoded into different formats to support the variety of users' requirements, such as screen resolution. The encoded files will be stored in unique URLs. Typically, the server of progressive downloads is a stateless server, which results in a stream switching process where the users have to switch the stream to suit their preference. The traditional Youtube video is one of the examples of this scheme [16].

In contrast to the stateless system, Real-Time Streaming Protocol (RTSP) can be a stateful streaming delivery system where information of streaming sessions are recorded [17]. To adapt bit rate, the stream switcher (scheduler) of RTSP has to exploit the feedback messages that indicate the network condition or buffer occupancy level to select the appropriately encoded video where its bit rate suits the conditions the most [13].

Other adaptive streaming techniques are HTTP-based Adaptive Streaming (HAS) and Dynamic Adaptive Streaming over HTTP [18-20]. With these protocols, a video is separated into small segments of a few seconds. Each segment is encoded into many versions with different bit rates. The client starts a streaming session by downloading the manifest file which details the related information describing the video stream. The client then starts downloading the chunks using HTTP command. At this stage, the client estimates the network conditions and buffer occupancy, then selects the next appropriate chunk that suits the conditions the most. This mechanism continues as a loop to estimate bandwidth availability and then selects the next proper chunk until the end of the movie.

For scalable video streaming using HAS/DASH, there is some research that presents this issue. For example, research in [21] presented using multi HTTP/TCP connections to download

multi-layers of scalable video simultaneously. In [22], the research studied scalable video streaming in DASH on Software Defined Networks (SDN).

To sum up, the challenges in video streaming are related to many areas. Firstly, encoding techniques are the process to produce the appropriate video stream for delivery. Secondly, in the field of video delivery, we have mentioned about the progressive download, RTSP, and HAS/DAS. Thirdly, what we have discussed so far is how the protocols exploit the structural advantage of the encoded stream to combat the bandwidth fluctuation.

2.2 De-jitter Buffer Management

This section explains an important part of the video streaming system—the de-jitter buffer. This section is comprised of five subsections. Subsection 2.2.1, is the traffic profile that briefly describes the characteristics of traffic in the network that it is the cause of an unpredictable end to end delay. In subsection 2.2.2, the jitter or the variation of delay between two consecutive packets that arrive at the receiver will be explained. In subsection 2.2.3, the effects of jitter on the quality of the video will be explored. Subsection 2.2.4 will explain how the de-jitter buffer works.

2.2.1 Traffic profiles and QoS

In this section, we try to identify the pattern of traffic that causes the delay and congestion in a best-effort and share network. A flow of data produced by a user can be roughly categorized into one of these three profiles [23] as shown in Figure 2.4.

Figure 2.4 (a) shows the constant bit rate where the data rate does not change in time. Also, there is no peak rate. This type of profile is predictable, so if the network knows the requirement

of the application that generates the flow, the network can allocate the suitable bandwidth for each data flow.

In Figure2.4 (b) the variation bit rate is shown. The data rate changes in time. In this case, the peak data rate and average rate might be different. So, regarding bandwidth allocation, it is challenging to estimate the current data rate that has to be allocated. The data flow with variable bit rate traffic profile causes the jitter in another data flow. So, this type of traffic profile triggers the de-jitter buffer, as explained in section 2.2.3.

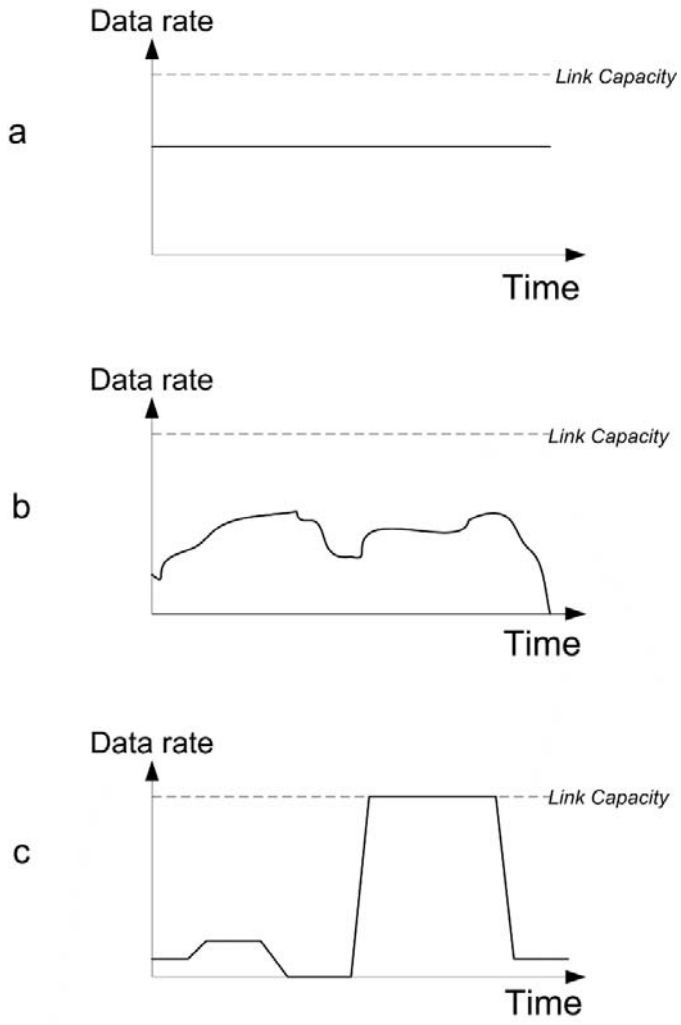

Figure 2.4 Traffic Profiles (a). Constant Bit Rate (b) Variable Bit Rate (c) Bursty

In Figure 2.4 (c) shows the bursty traffic profile. The data flow rate suddenly changes in time. The data rate may change suddenly from the lowest point to the peak in a very short period. The congestion is usually caused by this type of traffic profile. We can reduce the effect of bursty traffic profile by using the reshaping algorithm in an immediate network node (router).

The reshaping technique shown in Figure 2.5 employs the leaky bucket algorithm [24, 25] to smooth the peak of data flow. Assuming that there is heavy rain in a short period, we can smooth the flow of water by using a huge bucket that has a small hole at the bottom. The water leaks from the bucket at fixed rate while the rest of water is temporarily stored in the bucket. In a network, if bursty traffic arrives at a router, the queueing processor in the router will remove each packet from the queue at fixed rate, as shown in Figure 2.5 (b).

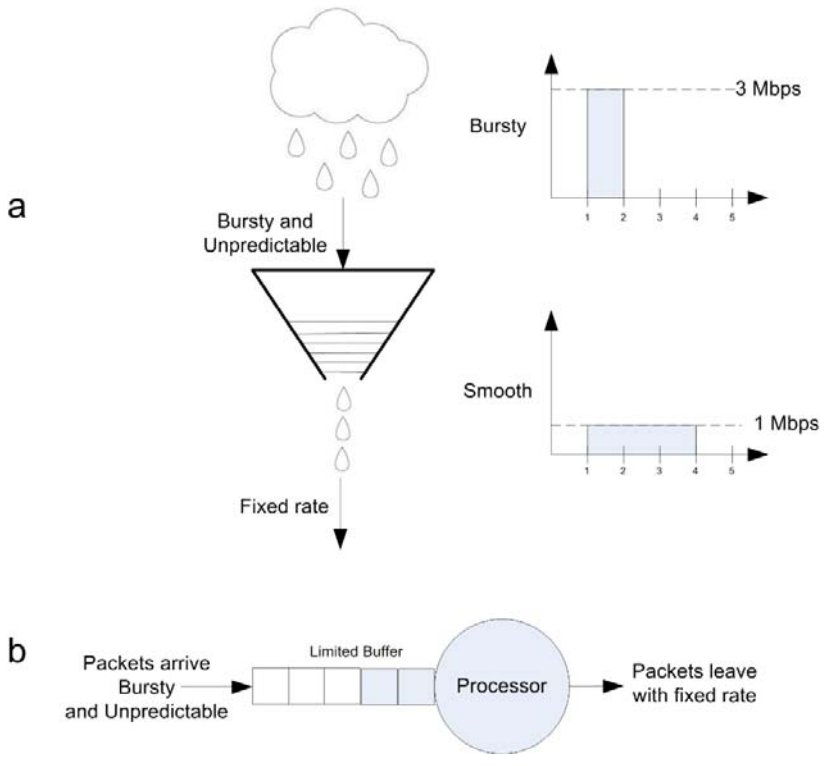

Figure 2.5 Concept of leaky bucket to shape traffic.

Please note that if the bursty flow is reshaped, the delay will happen. As shown in the right hand of Figure 2.5 (a), the bursty flow at 3 Mbps arrives at time 1, then it is shaped to 1 Mbps, but it uses 3 seconds to pass the queue, so the delay is increased.

In a shared network, QoS is the fundamental component to establishing fairness among users. This fairness is calculated in the sense that a user in higher classes gains better service than those in a lower class and users in the same class should receive services equally. So, the user in a best-effort class is more susceptible to traffic load in the network than the user in the higher class of services.

2.2.2 Jitter and Its Effect on Perceptual Video Quality

Jitter is the variation of delay between the two consecutive packets caused by queuing. It is possible to see jitter regardless of the congestion level [26, 27]. The jitter causes the degradation in Quality of Experience (QoE) either regarding video interruption or video quality impairment which affects viewer satisfaction considerably [28, 29]. Figure 2.6 represents the concept of jitter. Consider the data of Frame 1, if there is no jitter in the first bit of Frame 1, it should arrive at the receiver at time t_{r0}. So, the last bit of Frame 1 arrives at the receiver at t_{r1} . At the time of t_{r1} , it is too late for Frame 1 to be decoded because the decoding deadline is defined at t_{p1} . In this scene, we can say that the jitter presented in the network caused Frame 1 to arrive at the receiver later than its deadline and will never be decoded and displayed. The concealment might be required to repair the distorted video.

Research in [30] and [14] report that the buffering time (time spent on re-buffering) has the largest impact on user satisfaction. The second biggest impact on user satisfaction is initial buffering time or startup time.

If there are no variations of delay, the time between the two consecutive data will never change, the flow of video observed at the receiver is as smooth as when at the sender. On the contrary, jitter is still present. The communication statistic between two ISPs in California was observed by [31] and the result revealed that, especially during prime time, jitter was more than a hundred milliseconds and nearly 14% of test packets arrived at the destination after the deadline, with enough potential to distort the video streaming quality. The research in [32] developed the protocol that tries to reduce the effect of jitter to guarantee the quality of service.

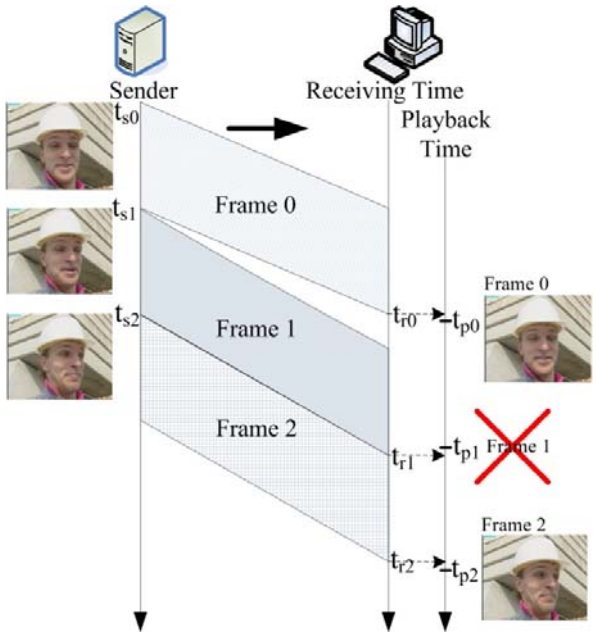

Figure 2.6 Jitter or variation of delay causes degradation in the continuity of video streaming.

2.2.3 De-Jitter Buffer Model

The function of the de-jitter buffer is to temporarily store video data before playback. At the starting point in time as shown in Figure 2.7 (a), the de-jitter buffer is filled until an amount of video data stored in the de-jitter buffer is higher than the threshold. Normally playback frame rate is constant, but video data arrival rate is varied. So, if video data arrival rate is greater than playback

rate, the number of video data stored is increased, as shown in Figure 2.7 (b). Conversely, the de-jitter buffer level is decreasing when playback frame rate is larger than video data arrival rate, as shown in Figure 2.7 (c). In the case of the buffer occupancy level being constantly decreased, this causes the buffer to empty, and then playback will be interrupted.

If video data arrival rate is greater than playback rate, the de-jitter buffer will grow continuously. The effect of the continuous growth of de-jitter buffer is an overflow. Therefore, the video delivery protocol should also prepare the mechanism to handle buffer overflow.

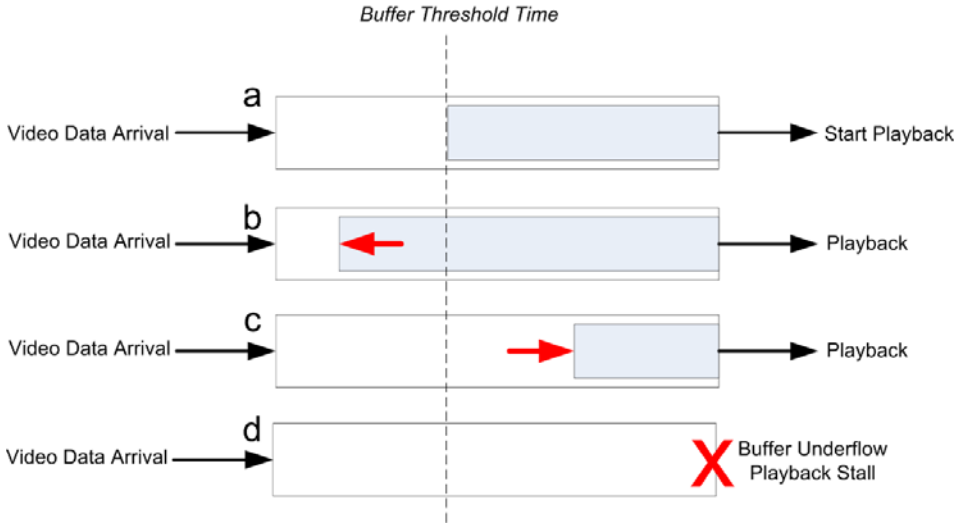

Figure 2.7 De-jitter buffer

(a) Start playback when the number of video data reaches buffer threshold

(b) The number of video data in the buffer is growing.

(c) The number of video data in the buffer is shrinking.

(d) There is no video data in the buffer.

As mentioned so far, the de-jitter buffer stores incoming video data temporarily. Figure 2.8 shows the buffer size and its ability to resist the jitter. Let's assume that the system is a time slot system marked with each time slot - t_1 to t_{10}. In Figure 2.8 (a), some slots have some packets arrive such as t_0, t_1, t_2, t_4, t_5 and t_8, but some slots do not have such as slot t_3, t_6 and t_7. Please note that in slot t_6 and t_7, no packets arrive but they are delayed and then arrive at slot t_8 in three packets. In this scenario, the stall events occur three times at slot t_3, t_6 and t_7 because there is no initial buffer.

In Figure 2.8 (b), let's assume that the initial buffer is 1 slot which is indicated by B(t) = 1 slot where B(t) is initial buffering time. The first decoded frame is played at slot t_1. The result is that the stall event at slot t_3 disappears, compared to the case in Figure 2.8 (a). However, B(t) =1 cannot absorb the stall events at t_6 and t_7.

When B(t) = 3 slots as shown in Figure 2.8 (c), all stall events disappear. We can notice that the more initial buffering time, the ability to absorb the jitter is increased. However, it has to exchange with the startup time.

De-jitter buffer would be large enough to avoid video streaming interruption. In Figure 2.7 (a) the playback will start playing the first frame when the buffer occupancy level reaches the threshold level. The important question is what is the appropriate threshold? The example of a threshold being too large is in the case of downloading the whole video (Figure 2.1). The study in [33] presented the method to calculate the approximate size of de-jitter buffer at the given underrun probability.

From the fact that the best-effort network cannot guarantee the minimum available bandwidth it becomes the function of application layer protocol to calculate and estimate the bandwidth level. Some bit rate adaptation algorithms employ the de-jitter occupancy level as well as other network channel conditions to adapt sending bit rate. The study in [34] proposed bit rate

adaptation using buffer status and estimated a statistic of bandwidth availability. The research in [35, 36] solely uses buffer status to adjust bit rate in the steady state. The study in [37] presented that the accuracy of throughput estimation is limited, therefore it should not be employed to indicate the throughput. Thus, the research proposed a low-latency bit rate adaptive using buffer occupancy. Research in [38] proposed the adaptive bit rate algorithm to reduce the oscillation of the sending bit rate by using the buffer occupancy level only.

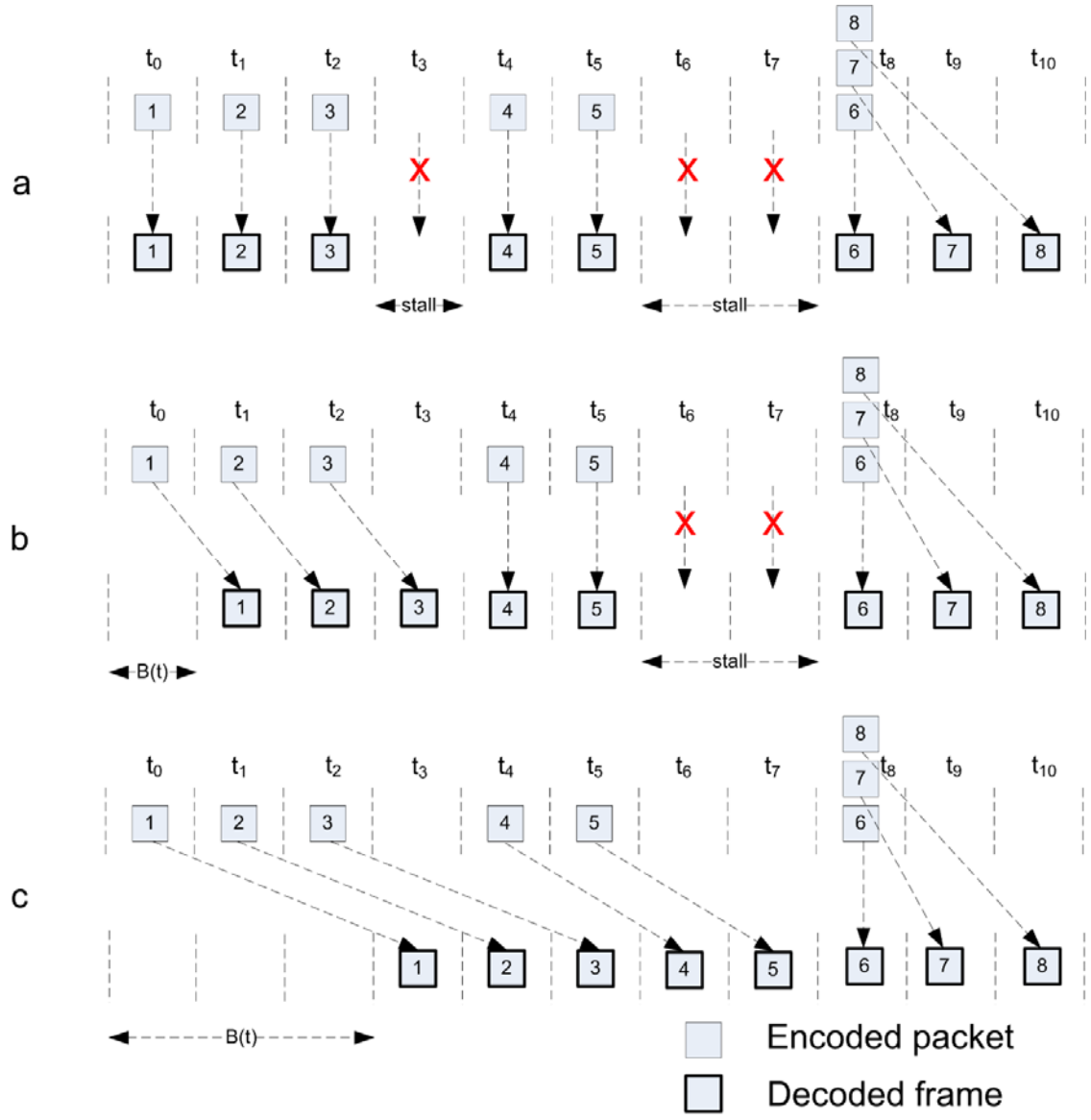

Figure 2.8 De-jitter buffer and its ability to absorb jitter.

2.3 Video Scheduling Algorithms

2.3.1 Scheduling and Packet Selection in Video Delivery

In a video delivery system, the scheduler is a strategy to order the sequence of packets and to allocate time used by each packet to resources of transmission. There are many objectives of the scheduler such as maximizing throughput, minimizing overall delay, ensuring that the video data arrives at the destination before the deadline or to guarantee fairness. It depends on system requirements: for example, in real-time video delivery the sender's scheduler should minimize the percentage of the frames that miss the decoding deadline, or in multicast video delivery, the scheduler might have to guarantee the equality of video quality appearing among users in the multicast group, etc. For packet selection, this is the process of choosing or discarding a packet from a set of packets. Figure 2.6 shows the diagram of packet scheduling and selection. We can notice that after the selection stage, some packets are discarded. Thus, the bit rate required for delivering the data is reduced. After discarding some packets, the rest of the packets will be reordered and allocated to the assigned channel.

There is considerable research in video delivery relating to congestion resistant. The research in [39] proposed a scheduling algorithm that mixes layer-based and frame-based to combat the congestion. When traffic is congested, layer-based is then utilized. However, if the traffic availability is stable, frame-based is employed. The research in [40-42] proposed a packet selection scheme to reduce the bitrate of video stream in case of low bandwidth; moreover, these packet selection algorithms attempt to minimize the distortion of the video.

The research in [43] developed a framework in multi-user scalable video delivery that maximizes the overall bitrate assigned to users and minimizes the difference in video quality appearing among users. The research in [44] proposed a strategy to improve the overall resource

utilization in LTE network and gain the quality fairness among users depending on the channel conditions.

The study in [45] regarded scalable video delivery in multicast developed multicast scheduling which assigns frequency resources to each subgroup in the multicast group. The results show that this scheduling algorithm achieves higher channel utilization and guarantees video quality. To maximize video quality in scalable video multicast by considering channel condition, resource availability, user requirements and power consumption, the research in [46] proposed the method of probing multicast channels by sending probing frames and then evaluating the channel conditions using the SNR of the frame. The scheduling and resource allocation applied a binary integer programming technique to match the video data with a suitable channel.

In [47], to maximize the total video quality in live video delivery, the research utilized the concept of effective capacity for statistical delay bounds to develop the scheduling algorithm and resource allocation policy. The study presented by [48] proposed a delay-aware scheduling algorithm that gives the highest priority to the highest delay packet. The analysis of delay-aware and capacity-aware bitrate allocation in UXP stream was presented in [49]. The study in [50] proposed a packet selection scheme which selectively drops some packets to reduce bitrate. This packet selection scheme facilitates the delay-aware video delivery over downlink OFDMA and minimizes its transmission power. The delay-aware and priority-based packet scheduling for video delivery in a mesh network using the learning method was proposed in [51]. This packet selection algorithm can estimate the possibility of a packet that might violate delay bound, the packet which is unlikely to arrive at the receiver before the deadline is dropped.

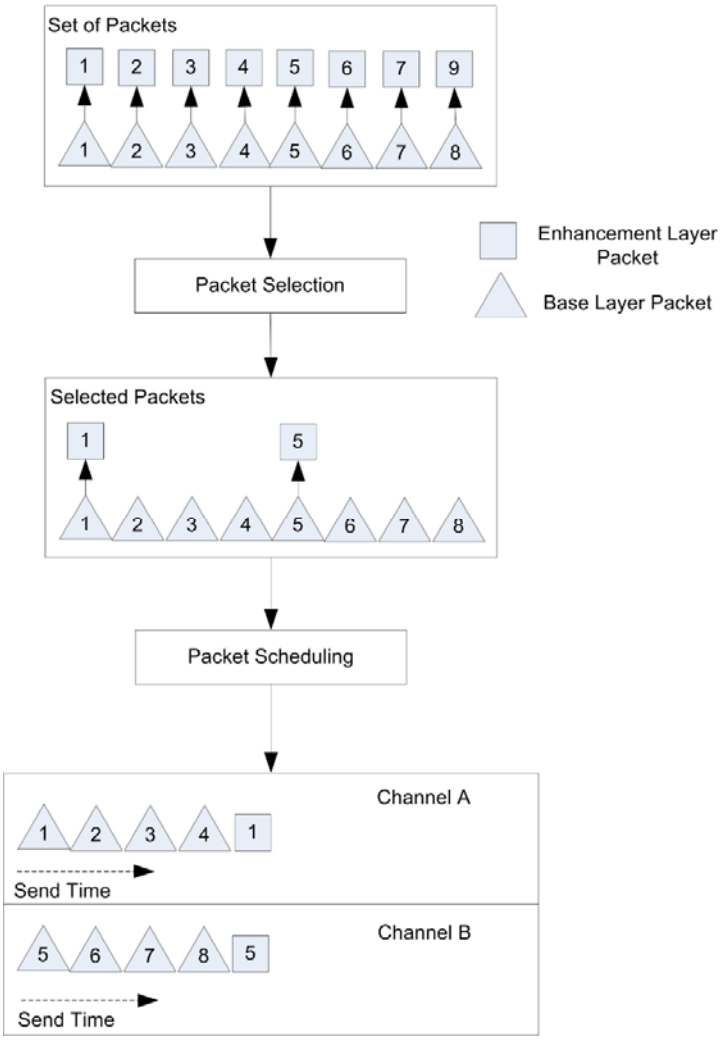

Figure 2.7 Overview of packet scheduling and selection.

In dense wireless networks, caching popular video data on users' device through device-to-device can achieve throughput scalability. Figure 2.8 demonstrates the basic mechanism of scheduling and resource allocation, which is comprised of a pair matching stage and channel allocation stage, in device-to-device networks. The research in [52] proposed the scheduling algorithm that was able to avoid the collision in device-to-device networks by applying graph theory to discover the conflict links to which the packets were not scheduled at the same time. The research in [53] utilized playback buffer length which indicates the probability of a stall event as

a key QoE index for which the scheduler can use to make the decision on what video quality level will be adapted, and an appropriate channel will be assigned. Another study relating to scheduling and resource allocation in high user density wireless network was proposed by [54]. The scheduling takes the packet delay and obtained transmission rate into consideration to match the pair of devices and allocate the suitable channel.

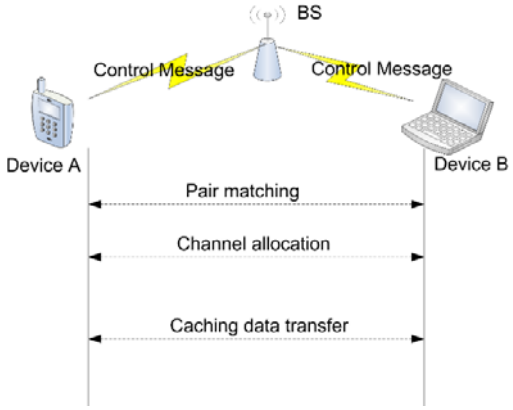

Figure 2.8 Mechanism before transferring caching data in device-to-device networks.

Another channel-aware scheduling algorithm was proposed by [55]. The scheduler evaluates link conditions and then selects the most suitable channel for transmitting data. In the transmission phase, the scheduler employs the concept of round robin to allocate a time slot of the selected channel to each user. This video delivery system obtains gains in both aggregate throughputs and per user throughput. The results indicate that video quality is improved significantly, especially for the nodes located at the edge of the cell.

In [56, 57], the playout buffer status can be taken into account to determine the packet prioritization process in scheduling algorithms. In serious congestion that causes the playout buffer level to decrease and finally underflow, the scheduler will provide the priority to the lowest temporal level of the base layer.

2.4 Scalable Video and its flexibility

2.4.1 Bandwidth Usage Flexibility

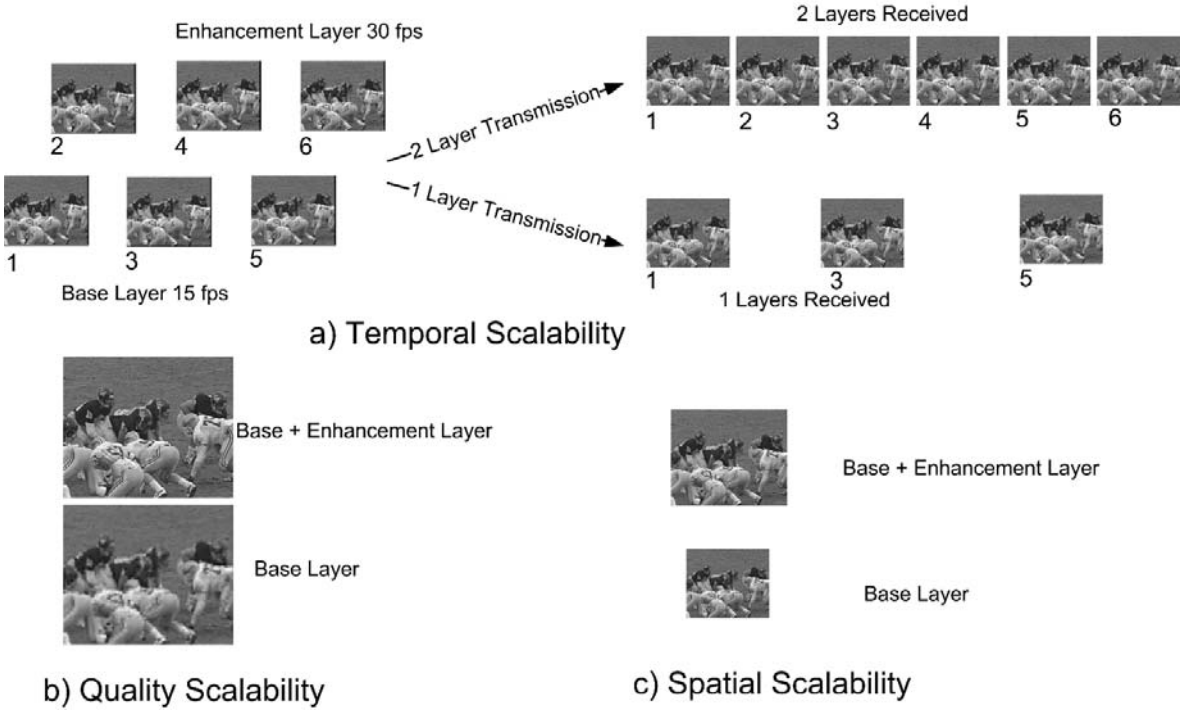

Figure 2.9 Temporal, Quality and Spatial Scalability

The scalability in scalable video stream can be categorized into three groups which comprises of temporal scalability, quality scalability and spatial scalability as shown in Figure 2.9 a), b) and c), respectively. Reducing bandwidth usage can be achieved by either reducing frame rate, quality of each frame or resolution of each frame. In this regard, scheduler will make decision on whether only base layer or base layer with enhancement layers that would be sent. Note that this process can be done without transcoding.

By exploiting the elastic structure of scalable video, there has been previous research conducted regarding scalable and its capability to be adapted to suit both bandwidth availability and to be QoS friendly video stream [58-60] because the scheduler can gradually increase bit rate of the scalable video stream to reduce the bursty traffic. Increasing bit rate gradually can be

achieved by slowly increase the number of enhancement layers or parts of an enhancement layer to the departure stream. In addition to adjust the bandwidth usage, other aims of dropping some enhancement layers or parts of enhancement layer is to safe power of display device [61, 62].

2.4.2 Unequal Protection

Since scalable video stream is composed of multi-layer with different importance, unequal error protection is applied to protect the essential parts of the video stream. For instance, [63] uses the forward error correction (FEC) technique to provide the highest protection level to the base layer rather than using only physical FEC. Moreover, in addition to provide protection to loss or error, the scheduling algorithm can provide protection to the packets to resist the variation of end-to-end delay. There are a number of research such as delay-aware scheduling algorithms that have been mention in section 2.3.

2.5 Related Topics in Video Streaming

2.5.1 Mobile Ad Hoc Networks and Video Delivery

A Mobile Ad-hoc Network (MANETs) is a self-organizing network without any fixed network equipment. The most important characteristics of mobile ad hoc networks are the dynamically changing network topology and the number of mobile users varying rapidly over time. As a result, data distribution over mobile ad hoc networks becomes one of the most challenging research topics, especially when one sender sends real-time multimedia data to many receivers.

The concept called multicast has been proposed to solve one-to-many data distribution. The multicast can be implemented in two possible ways. Firstly, it is known that an IP layer multicast that router duplicates data before sending to receivers. However, in an inter-domain

multicast, this scheme is not widely available because many ISPs are still reluctant to provide multicast routing on their routers. Secondly, the multicast concept can be implemented in the application layer. Over the last decade, many researchers have proposed the notion of application layer or overlay multicast instead of the IP layer multicast to solve this problem [64] [65]. Although the overall performance of overlay multicast is lower than IP layer multicast, its flexibility is the main advantage. The overlay multicast gives an opportunity to any mobile equipment in mobile ad hoc networks or even wired-network computers to communicate in a group communication scheme without extra networking equipment setup.

An overlay multicast is a virtual multicast using a P2P mechanism for forwarding packets compose of two main types: mesh-based and tree-based overlay multicast. Mesh-based or an unstructured overlay multicast system use a routing algorithm, for example, Forwarding Group Multicast Protocol (FGMP) [66], to route the packets to the destinations. On the other hand, the tree-based overlay multicast, such as MAODV [67, 68] or LGT, builds a tree structure before delivering data from the sender to the receivers. Although MAODV is a tree-based dissemination, its concept is on a share tree which may be highly affected by the quality of multimedia when the tree is fragile because both data and its compressed part are sent on the same route.

The multimedia applications over mobile ad hoc network using tree-based overlay multicast have a more predictable latency than those which use mesh-based overlay multicast [69] because, along the path from the root node to the receivers in tree-based, all packets are sent on the same route. Thus, the tree-based overlay multicast is more suitable for real-time multimedia distribution. However, in the case of tree fragility, the latency increases dramatically. An efficient protocol should recover the tree as fast as possible. The tree that lacks robustness is a major disadvantage [70] and tree construction protocol should be considered.

When the new factors have been proposed, some of them succeed in improving overall performance by using global positioning as one of its parameters to route data from a sender to receivers[71]. Other factors and appropriate algorithms to build the paths/trees would be considered, such as stability of the nodes in the tree.

Furthermore, in a unicast video dissemination scenario, some coding techniques with an appropriate structure can be used to correct the error as a result of packet loss when the first path is fragile. In the cast of H.264/AVC video on mesh P2P structure, the MDC is an interesting coding technique that was investigated by [72]. The bit stream of H.264/AVC video is split into two flows and will be sent to a different path. [73] has also suggested splitting the bit stream of the scalable video that should be observed. In this thesis, Chapter 6 explores multiple servers streaming.

On the other hand, from the last few years, VANETs became one of the most active areas in computer networks. Many data dissemination strategies have been shown [74, 75]. The research proposed an algorithm, structure, and protocol to send and receive data packets efficiently. The challenge of VANETS research rather than other wireless ad hoc networks is that the members of nodes in VANETs move rapidly compared to the speed of a normal human walking. The challenge of VANETs comes from a large number of variables that would be taken into consideration such as vehicle speed, density, and topology of the road [72, 74, 76, 77]. However, many advantages of 802.11p may be affected positively on the performance of the distribution tree because its coverage distance will reduce the probability of tree fragility when compared to its family [78].

2.5.2 Multipath One-to-One Video Delivery

In video transportation in mobile ad hoc networks, a link failure is a major cause of video quality degradation. Ensuring stability during the transmission of video is impossible in the link failure. Creating a backup link is an attractive option for this situation. It is hoped that both the primary and backup link is not broken simultaneously. In [79], the researchers build a multipath

transport infrastructure which split RTP flow into many sub-flows to resist transmission error and link failure.

A considerable amount of literature has been published on multipath one to one video dissemination, such as in [80, 81], who proposed the original concept of protecting base layer packets in base layer paths by using selective ARQ. The receiver will send the ARQ packet back to the sender to inform the sender regarding base layer packet loss. When the sender acknowledges the loss of base layer, it will retransmit the base layer packet through the enhancement layer path. This research also suggested the mechanism called reference picture selection (RPS) which assists the encoder to adapt the encoding process up to the prior lost frame. The two mechanisms have introduced the idea of using feedback packets from the destination to adapt the encoding process and to choose an optimized path for a base layer.

The research in [82] presented scalable video dissemination with a multipath strategy using the bandwidth available as a parameter to choose the suitable path. In [83], the researchers use feedback packets to estimate loss probability of the path. Then, the result from estimation will be used in the reference frame selection process which will be sent to protect error propagation at the decoder side. The research in [2] utilized quality of experience (QoE) receiving from users to select the best performance path. However, it is not clear in the practical application how they received an accurate feedback from the user.

2.6 Chapter Summary

This chapter mentioned the background literature and overview of the related issues regarding video streaming, buffer management, scheduling algorithms, structure of scalable video, and also mentioned about other related topics. This chapter attempts to focus on the root of the problems and give background knowledge in video streaming. A number of researches are

mentioned and discussed especially, the delay-aware scheduling and techniques which inspire the works in this thesis.

Chapter 3
Weighted Multi-playback Buffer Management for Scalable Video Streaming

3.1 Introduction

In recent years, the percentage of video data on the Internet has increased rapidly compared to other forms of data delivery. By 2017, almost a million minutes of video data will be transferred across the Internet every second, and all forms of video delivery over the Internet will increase to between 80 and 90 percent of all internet traffic [84]. Consumer Internet video flow, which excludes P2P video file exchanges, will be up from 57 in percent in 2012 to 69 in 2017 [84]. These figures show the important role that video transport has both today and in the future.

The Internet is a best-effort network, a network that does not guarantee the quality of service and one in which we cannot expect a minimum throughput or delay [85]. For video delivery, a best-effort network makes it impossible to guarantee a Quality of Experience (QoE) [86]. Video streaming in an insufficient bandwidth may cause discontinuity of the video flow since the transmitted video frames reach the destination after the expected playout time.

Scalable video coder provides layered video encoding/decoding and is designed to suit video broadcasting in best-effort networks [87, 88]. The scalable video allows the video data to be encoded into several layers. The video sender can select the number of encoded video layers to send to the receiver [89]. If there is plenty of bandwidth available, all the layers can be sent, but if bandwidth is low, the number of layers can be suitably reduced. The sender reduces the size of the transmitted video by sending only the most important layers and neglecting the higher

layers. In a fluctuating bandwidth, the SVC can tailor videos to fit and avoid the "it works or it does not work" [87] situation.

In this research, we propose an algorithm that provides unequal levels of data in the playback buffers. The structure of our algorithm is presented in Figure 3.1.

The rest of this chapter is organized as follows: in section 2 the related work is presented. In section 3 we present the scalable video rate adaptation and the operational points. In section 4, the proposed algorithm: the Weighted Multi-Playback Buffer (Weighted MPBuff) and related scheduling algorithm are introduced. The simulation setup and performance analysis are presented in section 5. The last section presents the conclusions.

3.2 Related Works

Much research has focused on the scheduling and selection of scalable video data to produce the best QoE. The research in [39] proposed a hybrid method of frame-based and layer-based scheduling for resilience to bandwidth fluctuation. The work in [40] selects the most important scalable video data to improve the PSNR. However, these algorithms do not consider the amount of data in any of the playback buffers and treat each of them with equal probability. Our algorithm maintains an unequal amount of data in the receiver buffers by intelligent reordering. In doing so, our algorithm provides greater fault tolerance to network congestion.

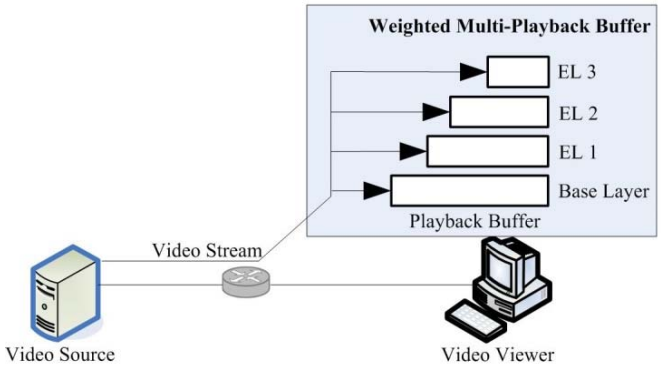

Figure 3.1 The architecture of weighted MPBuff management.

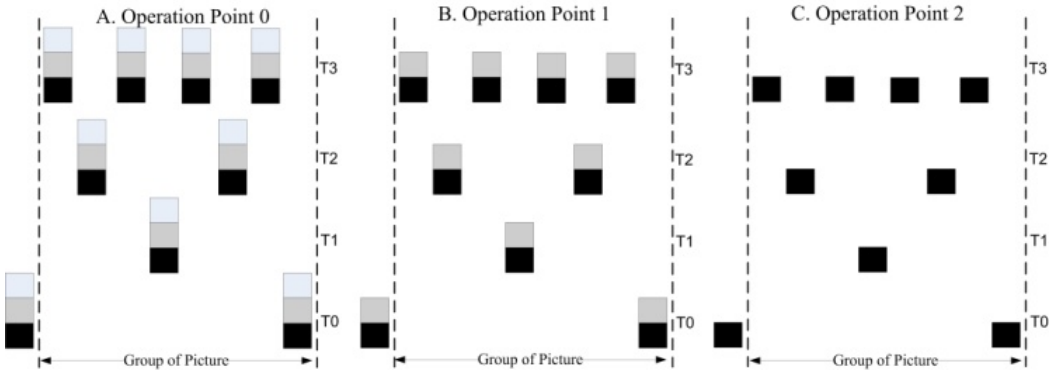

Figure 3.2 An example of possible operational points.

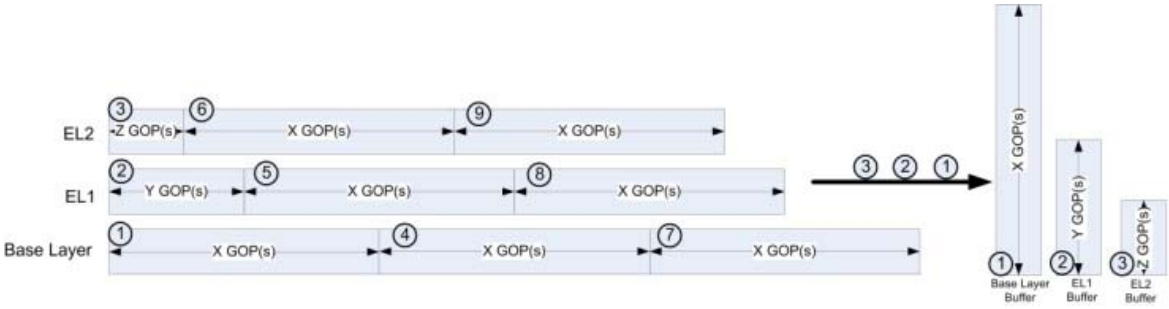

Figure 3.3 An overview of the scheduling scheme utilized by sender to support weighted MPBuff algorithm.

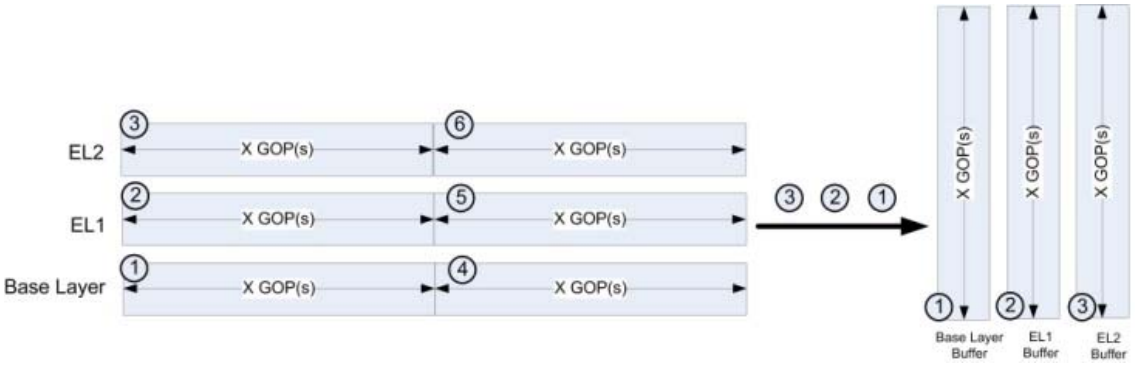

Figure 3.4 An overview of equal MPBuff Scheduling.

3.3 Scalable Video Rate Adaptation

This section describes the basic concept of adjusting the rate of an scalable video stream. The scalable video provides multi-layer encoding and decoding which allows an encoded video to be delivered with varying spatio-temporal qualities. The scalable video stream is comprised of one base layer and none or more enhancement layers. The base layer represents the lowest spatio-temporal quality of the video. If sufficient bandwidth is available, the enhancement layers are provided along with the base layer. It should be noted that the receiver cannot decode a frame if the base layer for that frame is discarded or fails to reach the receiver before the decoding deadline.

In order to achieve multi-level rate adaptation to fit the current bandwidth availability, one or more layers (not base) of the original encoded video can be dropped. In Figure 3.2 we assume that the video is encoded with a GOP 8 coding structure and is composed of one base layer plus two enhancement layers. Operational point 0 in Figure 3.2(a) shows no video dropped, the full spatio-temporal quality of the video is delivered. For operational point 1 in Figure 3.2(b), the top enhancement layer is dropped causing the bit rate of the video stream to be reduced. In operational point 2, only the base layer is permitted, with all enhancement layers being discarded.

3.4 Proposed Algorithm

In this chapter, Weighted Multi-Playback Buffer Management (Weighted MPBuff) is proposed with the continuity of the video and the associated QoE in mind [31, 90]. A viewer can detect the discontinuity of video easier than the spatial quality degradation. Our Weighted MPBuff and scheduling algorithm provides more protection to the lower layers compared to the higher layers. For example, the lowest layer or base layer will be given higher protection than the first and second enhancement layers. Weighted MPBuff expects that the probability of the

base layer playback buffer being empty will be the lowest among all layers. An overview of weighted MPBuff is presented in Figure 3.1.

To archive weighted protection, the sender needs to schedule the video data as shown in Figure 3.3. The number in the circle is the transmission sequence of the video. X , Y and Z denote the number of GOPs sent in each slot where $X > Y > Z$ build up an unequal buffer at the receiver. If $X = Y = Z$, the equal buffer is of the form shown in Figure 3.4. In our simulation, we compare the two schemes and their performances.

Assume that at time t after the receiver gets the first video data unit, the average length of the base layer's buffer is long compared to the other buffers as shown in equation (3.1). However, for the equal buffer scheme at time t after the receiver gets the first video data unit, the average length for all the buffers are equal as shown in equation (3.2).

$$\mathrm{L_{BASE}} > \mathrm{L_{EL1}} > \mathrm{L_{ELI}} > \mathrm{L_{ELI+1}} \tag{3.1}$$

$$\mathrm{L_{BASE}} = \mathrm{L_{EL1}} = \mathrm{L_{ELI}} = \mathrm{L_{ELI+1}} \tag{3.2}$$

where

- l_{base} is the average length of the base layer buffer.

- l_{ELi} is the average length of the enhancement layer buffer i^{th} where i > 1.

For the weighted MPBuff algorithm, the video buffer time ($v[i]$) of each buffer is calculated using equation (3).

$$\mathrm{v[i]} = \mathrm{t_h[i]} - \mathrm{t_c[i]} \tag{3.3}$$

where

- $v[i]$ denotes the video buffer time of the i^{th} buffer

- $t_h[i]$ denotes the highest timestamp of the i^{th} buffer

- $t_c[i]$ denotes the timestamp of the current playback time.

Figure 3.5 demonstrates an example of the three parameters used in weighted MPBuff and is given by equation (3.3).

At each time period, the receiver will calculate the video buffer time from equation (3) and report the parameter $v[i]$ for all playback buffers back to video source. After the sender receives the report $v[i]$ from the receiver, the sender will run the weighted MPBuff as in Figure 3.6. The results from the weighted MPBuff algorithm will be used to decide whether the video data unit in layer i+1 will be scheduled or not. For example, the sender adapts the transmission rate to fit to the bandwidth available by enabling layer i+1 to be sent if $v[i] > t_u^i$.

3.5 Simulation and Results

This section shows the simulation of the weighted MPBuff algorithm. The simulation is conducted in a rapidly fluctuating bandwidth using NS3. We compare the weighted multi-playback buffer scheme with the equal weighting generally used in SVC.

Figure 3.7 presents the dumbbell topology used for our simulation. The shared bottleneck link is constructed. The first pair of end nodes is dedicated to the video flow whilst the second pair is for background traffic. All link capacities are set to 2500 kb/s and link propagation delay is set to 1 ms. An ON/OFF UDP flow with an exponential random variable and a TCP flow have

been used as the background traffic. The mean of the UDP packet is set to 1200 bytes. We vary the ON time period between 0.25 to 1.5 seconds and the OFF time is constant at 0.5 seconds.

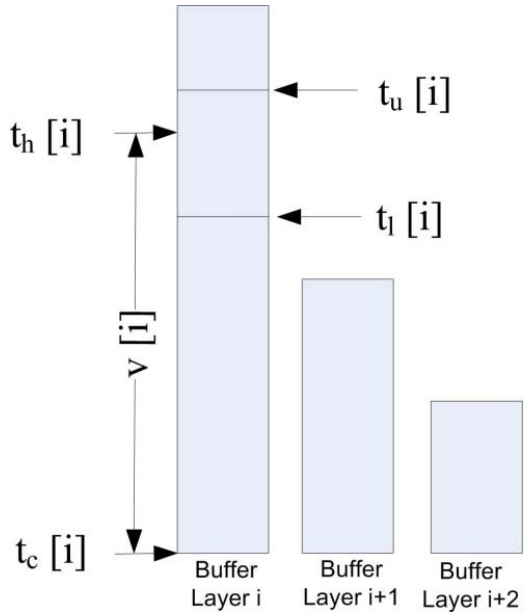

Figure 3.5 The definition of variables used in weighted MPBuff and equation (3.3).

Weighted MPBuff Algorithm:

----------*Initial Section*----------

-i = 0

-StopcheckBuffer = **False**

----------*Main Section*----------

Do:

If $v[i] > t_u^i$ **Then**

//allow layer i+1 to be sent
EnableLayer(i+1)

ELSE

If $v[i] \leq t_u^i$ **and** $v[i] \geq t_l^i$ **Then**
//do nothing
ELSE

//do not send layer i+1 and above
DisableLayer(i+1)
StopcheckBuffer = **TRUE**

ENDIF
ENDIF
i = i +1
Until ($i \geq MaxOP$) **or (** StopcheckBuffer is **TRUE)**

Figure 3.6 Weighted MPBuff Algorithm.

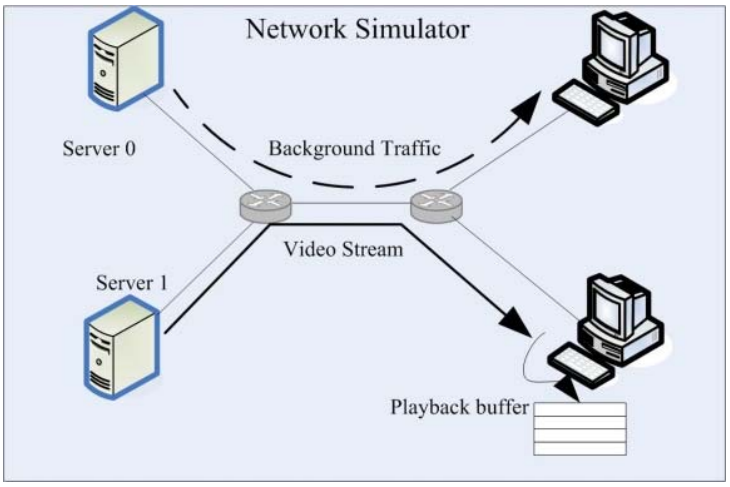

Figure 3.7 Network topology used in this simulation.

The five video clips used: harbor, foreman, city, soccer and mobile are concatenated. The combined video length is 1500 frames, CIF (352x288) resolution and the frame rate is 30 fps. We use a GOP of 8 frames. The JSVM reference software [91] is utilized to encode the video. The encoded video is composed of a base layer with QP equal to 38 and two quality layers (MGS) with QP equal to 35 and 33 respectively. We also further separate each enhancement layer into 2 layers to produce a 5 operation point video stream. The network simulation is set to 800 seconds for each run. The $\boldsymbol{t_u^i}$ and $\boldsymbol{t_l^i}$ of weighted MPBuff are presented in Table. 1 and the $\boldsymbol{t_u^i}$ and $\boldsymbol{\tau_l^i}$ of equal MPBuff is set to 2.4s and 2.0s respectively, spread equally over each buffer.

Table 3.1 Threshold $\boldsymbol{t_u^i}$ and $\boldsymbol{t_l^i}$ of weighted MPBuff

Buffer number	$\boldsymbol{t_u^i}$ (s)	$\boldsymbol{t_l^i}$ (s)
base layer	3.0	2.5
EL1	2.4	2.0
EL2	2.05	1.75
EL3	1.8	1.5
EL4	1.5	1.25

The simulation is analysed according to two performance metrics. First is the percentage of freeze frames and the second is Y PSNR. The cause of frozen frames is because information cannot be decoded on time, so the receiver is forced to conceal the missing information with information extracted from previous frames. Figure 3.8 shows the percentage of frozen frames at the receiver side. The weighted MPBuff outperforms the equal MPBuff by reducing the number of freeze frames, especially when the mean burst duration is varied between 1.0s and

1.25s. On average the weighted MPBuff provides 5.92 percent lower number of freeze frames than the equal MPBuff.

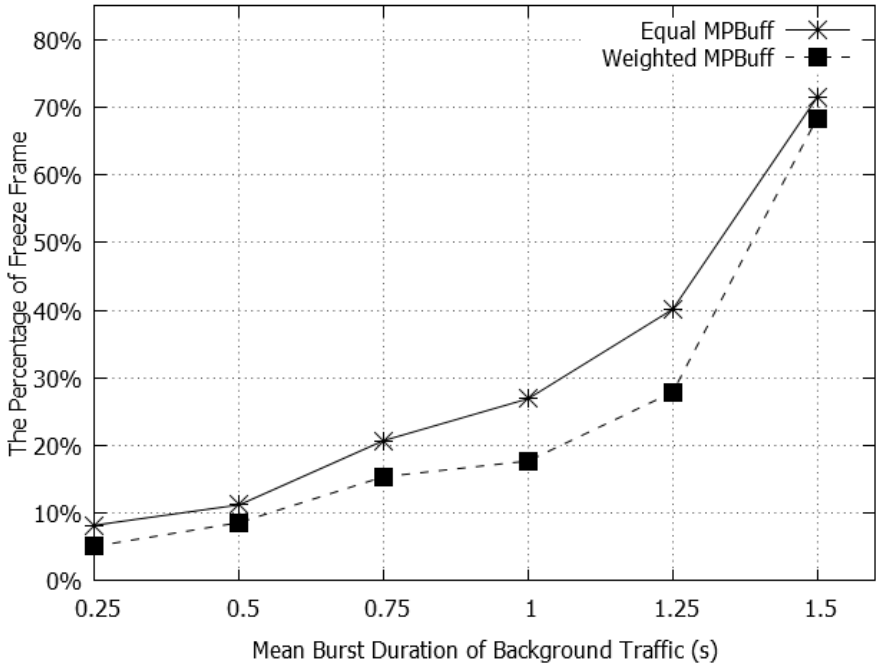

Figure 3.8 The Percentage of freeze frames when the mean burst duration of the UDP background traffic is varied between 0.25s to 1.5s.

The average Y PSNR is demonstrated in Figure 3.9 The weighted MPBuff gives a higher average Y PSNR than the equal MPBuff. The largest observable difference is when the mean burst duration of background traffic is varied between 1s and 1.25s. The Y PSNR of the weighted MPBuff is higher than equal MPBuff by 2.02 dB on average.

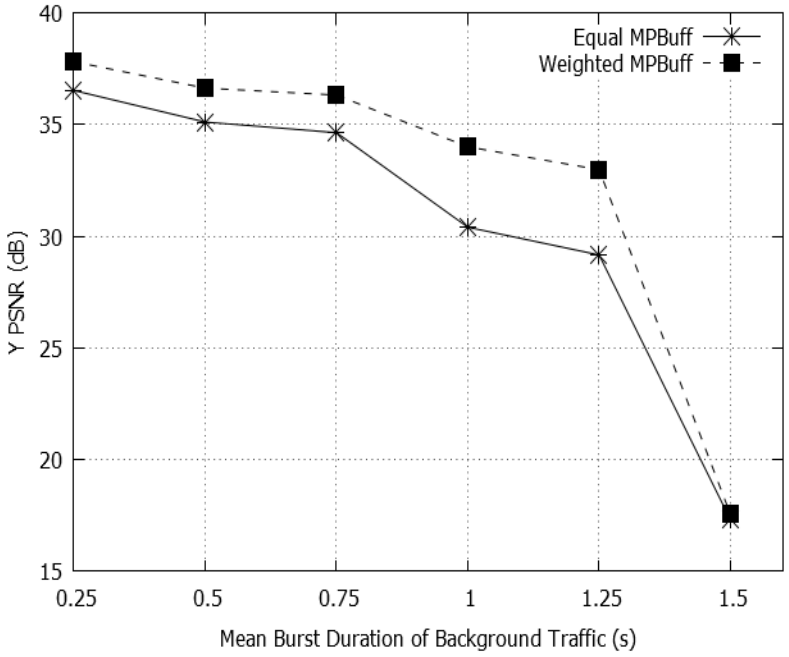

Figure 3.9 The average Y PSNR when the mean burst duration of the UDP background traffic varies between 0.25s to 1.5s.

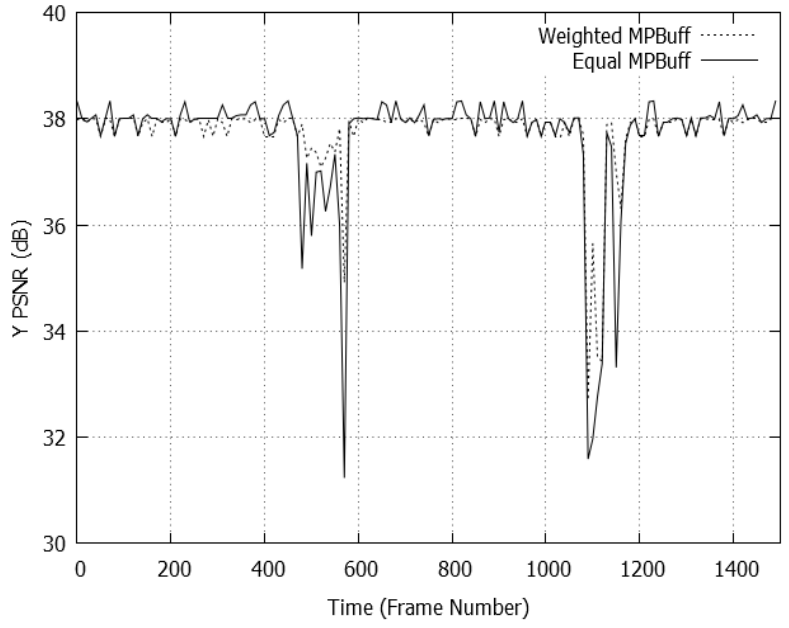

Figure 3.10 The Y PSNR when the mean burst duration of the UDP background traffic is 0.25s.

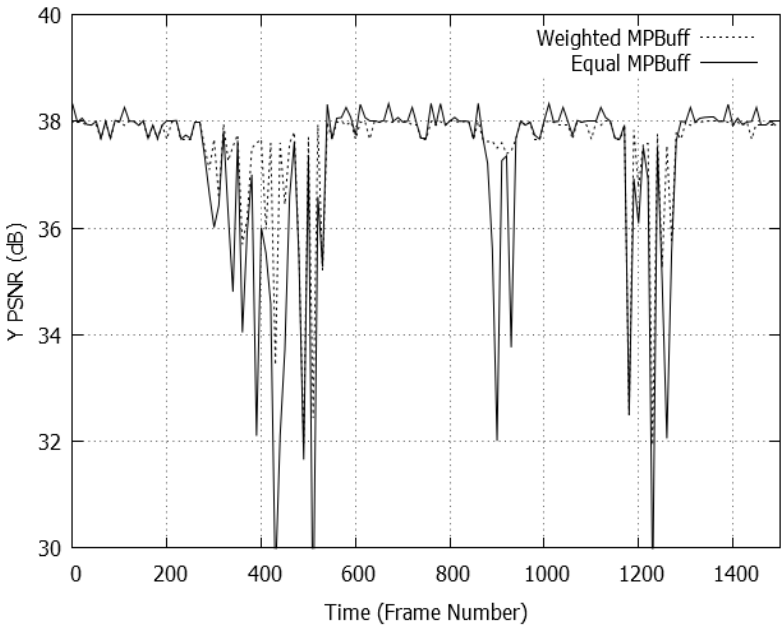

Figure 3.11 The Y PSNR when the mean burst duration of the UDP background traffic is 0.5s.

Figure 3.10 and Figure 3.11 show the Y PSNR when the mean burst duration of the UDP background traffic is set to 0.25s and 0.5s respectively. The two graphs clearly show that weighted MPBuff is more tolerant to fluctuating bandwidth than equal MPBuff.

3.6 Chapter Summary

In this chapter, the weighted multi-playback buffer algorithm (weighted MPBuff) is proposed. We compare the performance of our proposed algorithm with the equal multi-playback buffer (equal MPBuff) scheme. The simulation clearly shows that weighted MPBuff is more tolerant to bandwidth fluctuation. The weighted MPBuff gives a better level of continuity giving a lower percentage of freeze frames of 5.92% and giving a higher average Y PSNR by 2.02 dB on average.

Chapter 4
Continuity-Aware Scheduling Algorithm for Scalable Video Streaming using Look Ahead Scheduling Algorithm

4.1 Introduction

In recent years, the rise of consumer demand for the distribution and access of video content on packet networks has resulted in the development of a number of commercial offerings. Consumer electronic devices including set-top boxes, laptops and mobile devices have posed significant challenges to the video-related industry. Competition between service providers has resulted in a number of video standards being launched onto the market. Although network capacity has increased rapidly, a commonly encountered problem is bandwidth fluctuation. Indeed, an increase/decrease of bandwidth availability causes jitter or the variation of inter-arrival time between consecutive packets [26, 27, 31, 85]. Jitter results in an increase in the number of frames arriving at their destination after play-out time, *i.e.* with a deadline violation. The frames or packets which arrive at their destination after the deadline will not be decoded, and consequently, the average quality of the video is degraded [26, 31, 88].

Among the various video standards, the Scalable Video Codec (SVC), an extension of the H.264/AVC [87], has been designed to tolerate more adverse network conditions. The sender of scalable video can adjust the bit-rate to correspond to the available bandwidth without re-encoding. This approach has led us from “it works or it does not work” [87] to a more flexible video flow that may be shrunk to suit the available bandwidth. In other words, the

primary advantage of scalable video is its elasticity, where the video quality is tailored to the network conditions. In order to shrink a video bit stream, a particular portion of the data; a Network Abstraction Layer (NAL) unit can be removed from the encoded stream prior to delivery to the receiver. In the decoding process, the receiver is able to decode the bit stream if and only if all ancestor NAL units have been received and decoded correctly.

To decode any frame, at least one NAL unit from the base layer is required. For this reason, the base layer is the most important layer from the perspective of continuity and must be preserved.

This chapter gives more analysis and discussions to our previous study [92] which addresses the problem of discontinuity of video caused by the late arrival of NAL units. We introduce a novel sender side scheme which provides unequal jitter protection to a scalable video stream; called the Look Ahead Scheduling Algorithm (LASA). The scheduling algorithm tends to prioritize the base layer but not at the expense of degrading the overall PSNR.

The video stream could be smoothed by using playback buffer management in the receiver but at the expense of real-time interactivity. Our sender side scheme can also collaborate with a receiver based buffer mechanism and provides further improved performance. We exploit the advantages of scalable video by providing unequal look ahead to the SVC video layer.

The rest of this chapter is structured as follows. Section 4.2 is the related work. Overview and the pictorial explanation of LASA are presented in Section 4.3 and 4.4. Next, our proposed algorithm and related mathematical analysis are presented in Section 4.5 and Section 4.6, the simulation results are described in Section 4.7 and concluding remarks are given in Section 4.8.

4.2 Related Work

If there are no variations in delay, the time between two consecutive pieces of data will never change; the flow of video observed at the receiver is as smooth as at the sender side. If jitter is present, the time between consecutive packets is unknown prior to delivery. If real Internet traffic is observed [31] jitter can be of the order of hundreds of milliseconds which has the potential to distort the video. As a result of jitter, the perceptual quality is reduced and it has nearly the same effect as packet loss [93].

The primary mechanism to cope with playback interruption is to create a de-jitter buffer at the receiver [94, 95] where the incoming data are stored temporally before playback. The main problem with using only playback buffer to cope with bandwidth fluctuation is that in the event of high jitter, the end-to-end delay and playback delay will be high as well [96]. This chapter proposes a different approach, which is able to support each other boosting immunity to jitter.

Much work has been done on scalable video and its ability to adapt; either to shrink or stretch to suit the available bandwidth and user preferences [58, 97]. In addition to being flexible in unstable network conditions, the SVC is also capable of adjusting according to: the quality of the picture, the bit-rate or even the power limitations of the display device [61] . In [39], the collaborative streams share the same bottleneck link in a home network. Bandwidth probing estimates the available transmission bandwidth and the SVC flow is then adjusted to suit the channel capacity. The study in [98] uses two Transmission Control Protocol (TCP) connections to transmit one scalable video session with one TCP connection dedicated to the base layer and one TCP connection for all the other enhancement layers. Using parallel TCP connections gives more tolerance to congestion and jitter, however, the higher tolerance gives protection to the base and enhancements equally. The research in [99] presents an efficient video adaptation scheme in the Long-Term Evolution (LTE) networks using the distance

between user equipment and the LTE base station as a metric to reduce the size of the scalable video stream.

Layer-based and frame-based scheduling algorithms are mixed to suite the congestion and jitter level [39]. In the case of high congestion the layer-based scheme will produce better results. If the network is stable, frame-based scheduling is best employed. Alternatively packet selection schemes can be employed to reduce bit-rate, where the objective is to adjust the bit-rate by selecting the best group of packets that minimize the distortion [40-42]. The 3D scalable video scheduling in P2P networks was proposed in [100]. In this study, a client chooses the appropriate layers of scalable video along with a suitable depth according to the limited download bandwidth. The algorithm in this study also provides checking to guarantee the smoothness of the video by considering the playback deadline of each video packet in the base layer. The technique proposed in [101] is used to transmit scalable video streaming in MC-CDMA. To improve video quality, this study used partial channel state information (PCSI) to give different priorities to each layer of scalable video. The study in [102] proposed a technique using interleaving scheme to prioritize NAL units for real-time MVC video streaming. Other research on packet selection considers the status of the ancestor packets before sending the descendants [103, 104]. If an ancestor has not been scheduled, the descendant packet will be discarded. This method reduces the number of the descendant packets that are transmitted but cannot be decoded.

Due to the scalable video is a layered video scheme in which the video flow is composed of multiple layers, each with different importance. Unequal error protection can be applied to protect the more important layers using Forward Error Correction (FEC). The research in [63] first applied forward error correction (FEC) to provide prioritization in the network layer rather than more traditional physical layer. FEC is an effective method of protection but adds overhead to the stream. In [88], to increase the quality of video, the researchers give higher

priority to the more important layers using Reed-Solomon coding, this is in conjunction with priority-aware block interleaving (PBI) in the MAC layer. The research presented in [105] proposed an unequal recovery algorithm to lost packets which has higher retry transmission rates for the base layer compared to the enhancement layers. However, this algorithm needs a retransmission mechanism to archive the smoothness of video flow.

4.3 An Overview of Look Ahead Scheduling Algorithm (LASA)

This chapter proposes a new technique which uses round robin with unequal look ahead scheduling. The Look Ahead Scheduling Algorithm is named as LASA. By setting unequal look-ahead limits to each layer, unequal protection against discontinuity will be achieved. In this algorithm, the base layer is allocated a larger look-ahead limit than the other layers, the probability of a frame with deadline violation is decreased accordingly.

In order to describe LASA scheduling in detail, we compare our algorithm with frame-based and layer-based scheduling. In general, both frame-based and layer-based use traditional round robin scheduling. Frame-based scheduling algorithms schedule all enhancement NAL units which depend on the current base layer NAL unit. In other words, a base layer NAL unit and its enhancement NAL units which represent the same frame will be scheduled consecutively.

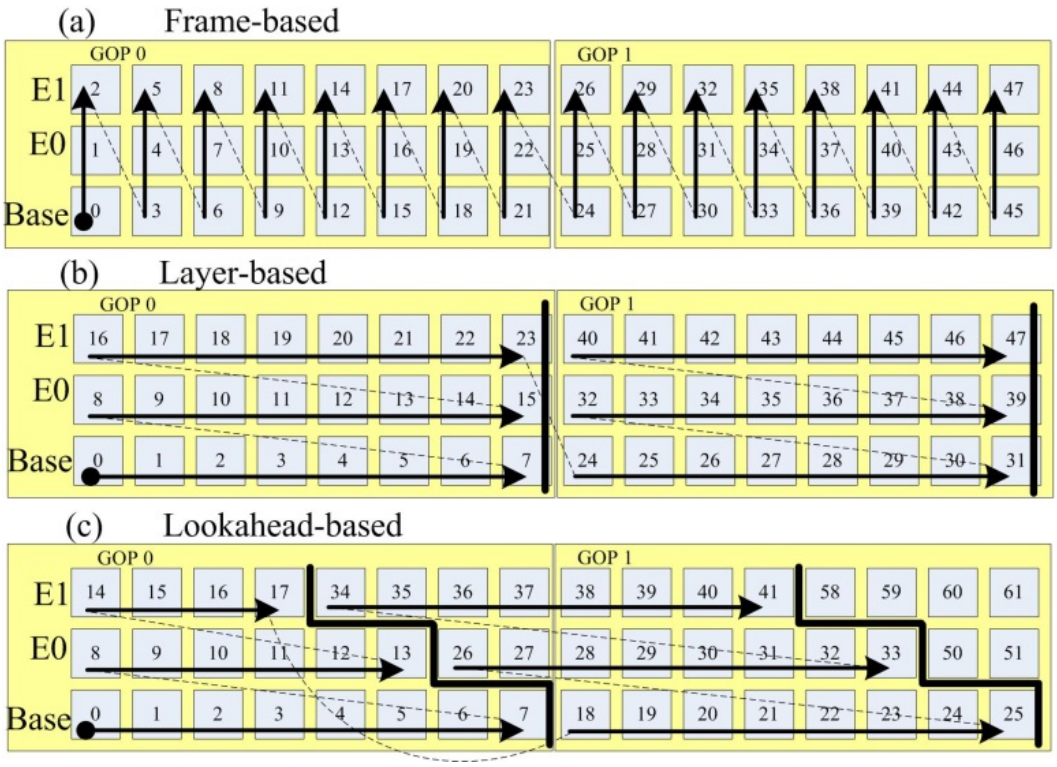

Figure 4.1 The concept of the three different scheduling schemes discussed.

(a) A frame-based scheduling algorithm;

(b) A layer-based scheduling algorithm; and

(c) The Look Ahead Scheduling Algorithm (LASA) scheduling algorithm.

In Figure 4.1, we assume that the scalable video which is composed of a base layer and two enhancement layers has a Group of Pictures (GOP) size of 8. The arrows and numbers show the order of NAL unit scheduling. The notion of a frame-based scheduling algorithm is shown in Figure 3a.

Alternatively, in a layer-based scheduling algorithm, the sending order proceeds along the horizontal axis until the algorithm reaches the upper bound marked by the dark vertical line in Figure 3b, after that the upper layer will be scheduled.

Our LASA scheduling algorithm is shown in Figure 3c. Similar to the layer-based scheduling algorithm, LASA orders the NAL units in the horizontal direction. The unequal bound, which is marked by the dark line, defines the upper bound of NAL units scheduled per layer. In the example shown in Figure 3c, the base layer is allocated with the most look-ahead value, whereas the top layer obtains the least look-ahead value.

4.4 Pictorial Explanation of LASA

This section gives an example of LASA by demonstrating the graphs of the number of frames that have been scheduled. The graphs are generated from LASA, layer-based and frame-based scheduling algorithms. This section will also demonstrate and give simple performance analysis of LASA in a graphical way which might lead to a better understanding of LASA.

4.4.1 Pictorial explanation of frame-based scheduling algorithm

Frame Number	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Layer 3	3	6	9	12	15	18	21	24
Layer 2	2	5	8	11	14	17	20	23
Layer 1	1	4	7	10	13	16	19	22

Figure 4.2 Traditional Frame Base Scheduling Algorithm

Figure 4.2., shows a basic traditional scheduling algorithm, here we call “Frame-based” since all layers of a frame are transmitted in consecutive order starting at the base layer (denoted as Layer 1 in Fig.1.). The order of scheduled video data units is marked with a number. We can notice that the first, second and the third of video data units marked with number 1, 2 and 3 belong to frame number 0. This is a frame-based scheduling algorithm.

The graph generated in this section is the result of a simulation of the three algorithms-LASA, layer-based and frame-based which is conducted under two assumptions. Firstly, one time slot will be dedicated to one video data unit. Secondly, both the propagation delay and transmission delay equals zero. The objective of this model is to build the simplest model possible to be the tool to compare the performance of LASA, rather than using only a mathematical model and network simulation.

In Figure 4.3, the dark blue line in the graph shows the number of frames that have been scheduled by the frame-based scheduling algorithm whilst the red line demonstrates the play-out time of the receiver. In this case, we give the play-out time to start at time slot number 4 where the frame number 0 is played out and at time slot number 4, frame number 1 is played out. The area of the dash rectangle block will be zoomed in and is shown in Figure 4.4.

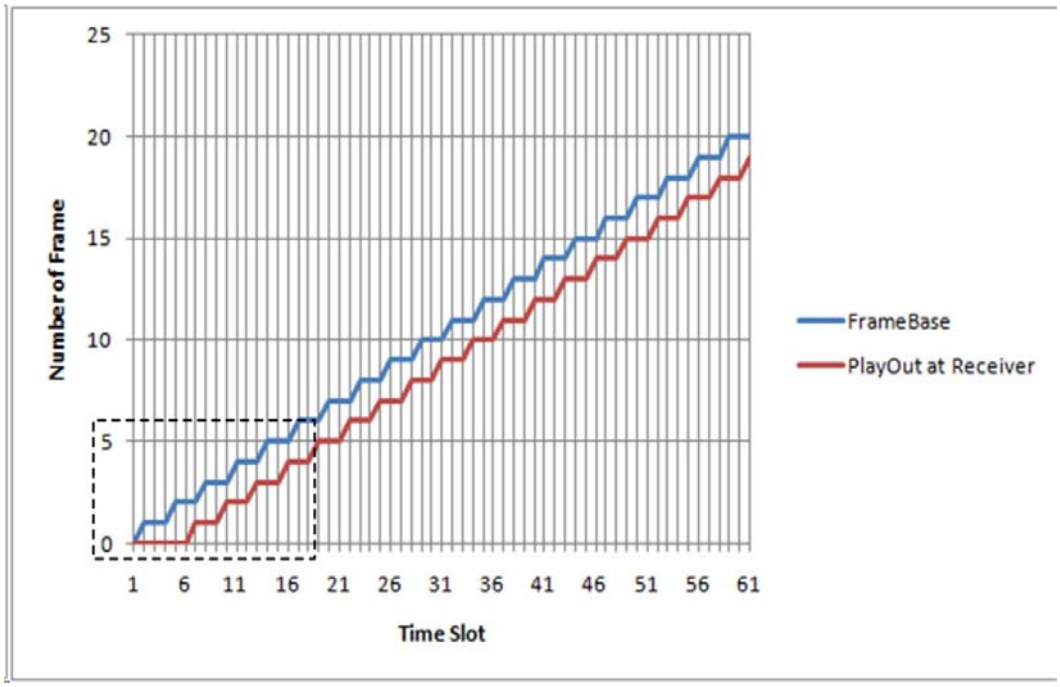

Figure 4.3 The number of frames which have been transmitted at a time slot.

In Figure 4.4, the time slot from vertical dash line A and B is 5 (13 - 8) and the same pattern repeats throughout our pictorial simulation. We will call the "time slot" before the frame is played out "time before deadline." We give the abbreviation to the term "time before deadline" as t_{BD}. In this case, $t_{BD} = 5$.

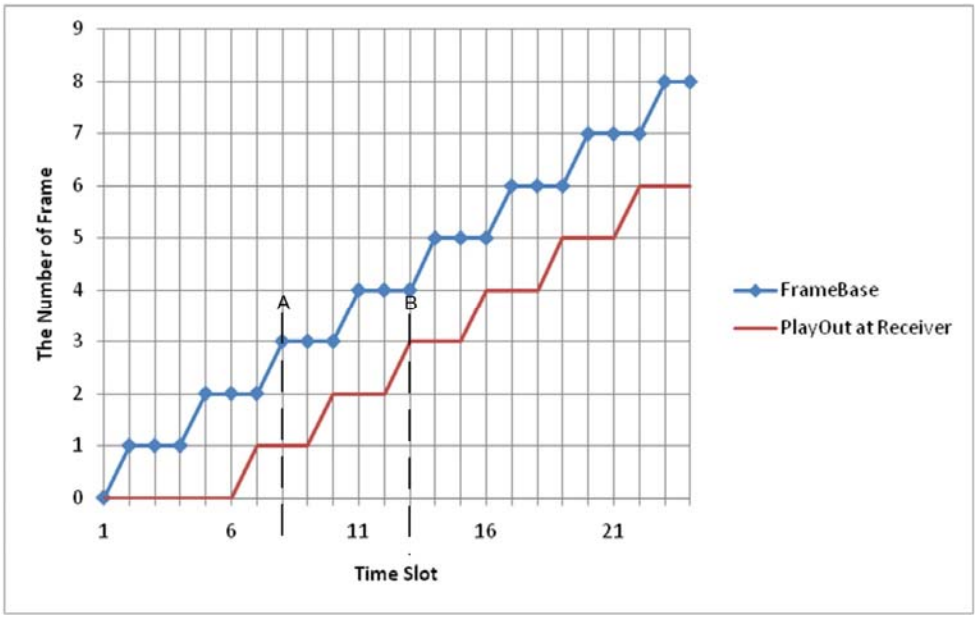

Figure 4.4 Time slot number 8 (marked by vertical dash line A) frame number 3 is transmitted and waiting to be played out at time slot 13 (marked by dash vertical line B).

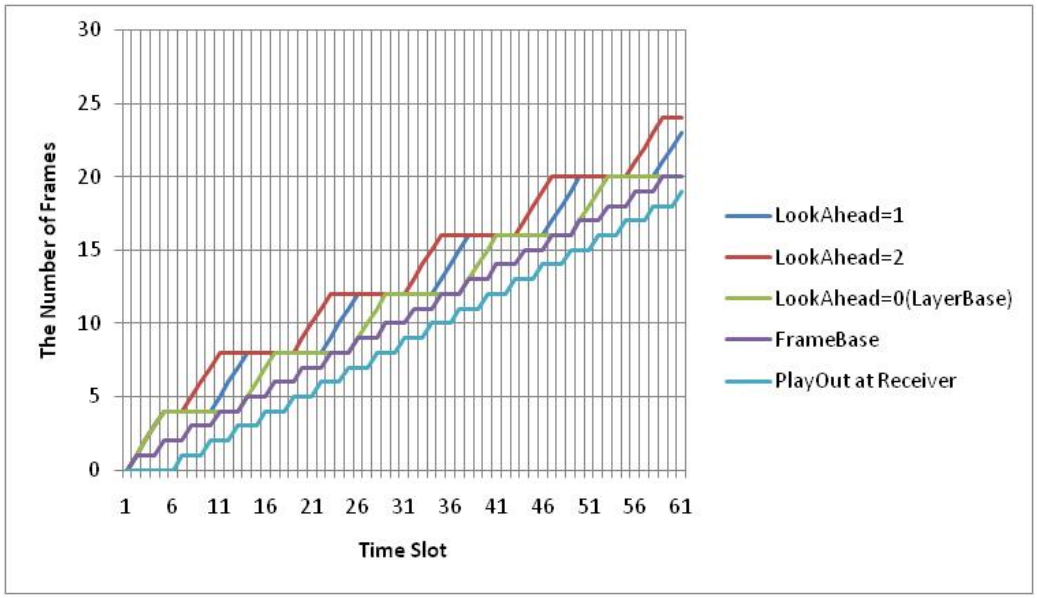

Figure 4.5. A simulation of LASA where the parameter equal to 1, 2 and 0 and Frame-based scheduling algorithm.

4.4.2 A Simulation of LASA and its rival

The pattern of LASA scheduling algorithm with LookAhead parameter equal to 1 is demonstrated in Figure 4.6.The pictorial simulation in this section uses a pattern to schedule video data unit and plot the graph of the number of frames transmitted.

Frame Number	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Layer 3	8	9	18	19	20	21	30	31
Layer 2	5	6	7	14	15	16	17	26
Layer 1	1	2	3	4	10	11	12	13

Figure 4.6 The pattern of LASA at Look ahead parameter = 1.

As shown in Figure 4.5, the pattern of the graph is repeated throughout our simulation, hence we will capture only the first 25 time slots and analyse the graph in detail as shown in Figure 4.7.

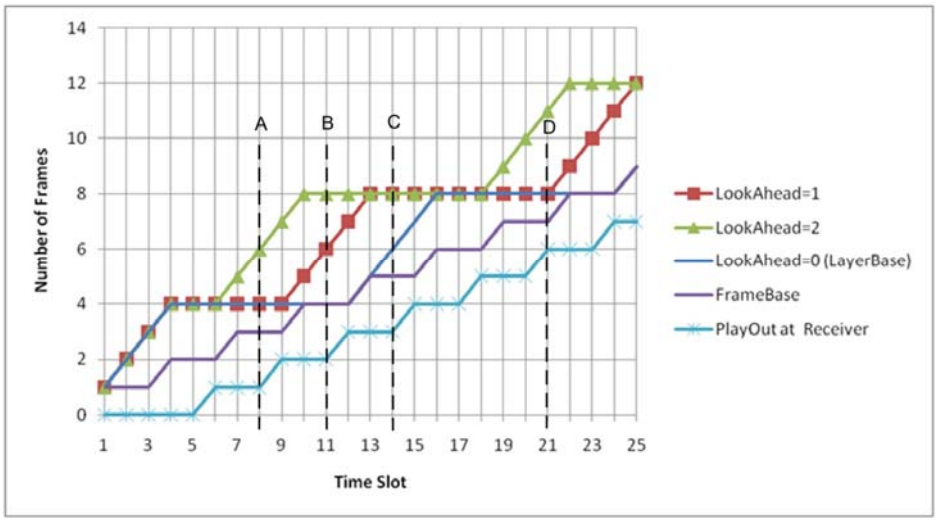

Figure 4.7 The first 25 time slots of a simulation of LASA

where the parameter equal to 1, 2 and 0 (layer-based scheduling algorithm) and Frame-based scheduling algorithm.

At time slot number 8 (marked by dash line A) LASA with LookAhead parameter equal to 2 can reach frame number 6 while LASA with LookAhead parameter equal to 1 can transmit frame number 6 at time slot number 11 (The difference between vertical dash line A and B is 3). In the same way, when the Look Ahead parameter changed to be 0, it can transmit frame number 6 at time slot number 1 (The difference between vertical dash line A and B is 3).

In this stage, we notice that every Look Ahead parameter changed to 1, the maximum difference between the graphs of LASA is 3. Given t_d is the maximum time slot difference from LookAhead parameter equal to 0.

Please note that when the LookAhead parameter of LASA is set to 0, the LASA behavior will be the same with the layer-based scheduling algorithm. So, the layer-based scheduling algorithm is a special case of the LASA.

$$t_d = \sum_{i=0}^{n} Li \quad (1)$$

Where Li is look ahead value of each layer.

We test the system by injecting a sharp drop of bandwidth. In Figure 4.8, the results show that frame number 5 of the frame-based scheduling algorithm at time slot 22 misdeadline causes video flow to discontinue. For the layer-based scheduling, although there is no frame misdeadline, the risk of frame misdeadline increase because in time slot 22 the receiver needs to play out video frame 6 but the layer-based scheduling algorithm has just finish transmitting the video data unit in time slot number 21. If the degradation of bandwidth is longer than 2 time slot, the layer-based scheduling will fail to schedule frame number 6 to the receiver before the deadline.

In this stage, we can conclude that every Look Ahead parameter changed to 1 had its ability to tolerate bandwidth fluctuation increased.

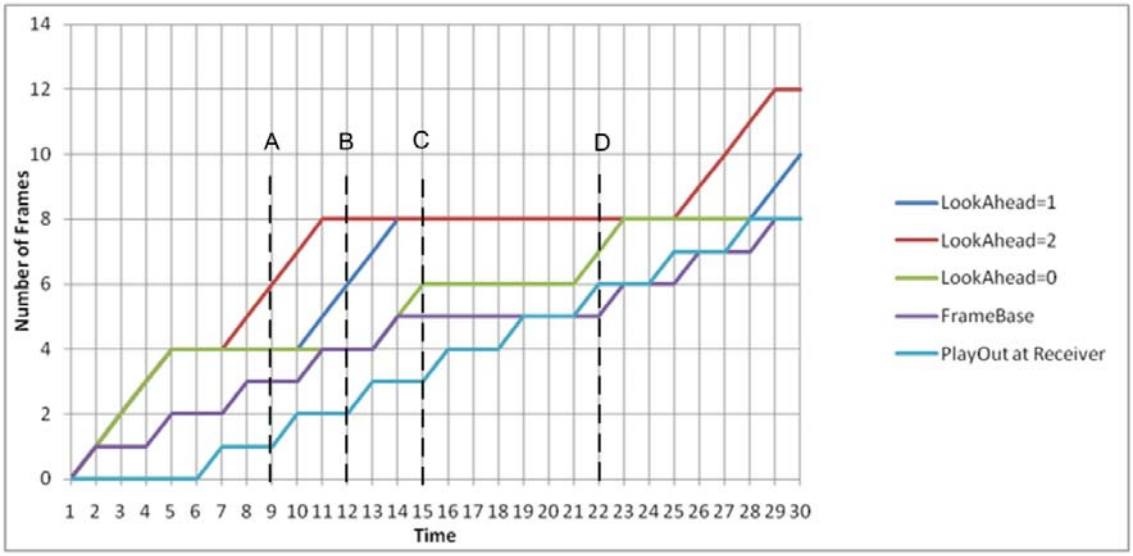

Figure 4.8 The first 25 time slots of a simulation of LASA where the parameters equal 1, 2 and 0 (layer base scheduling algorithm) and Frame base scheduling algorithm when bandwidth not available 5 time slot starting from time 16.

4.5 LASA Scheduling Algorithm and its Look-ahead Limit

In this section, we describe the essential parameters of the LASA scheduling algorithm. In Figure 4, there are two type of parameters look-ahead (Lbase, Le1 and Le2) and deltali.

L_{base}, L_{e1} and L_{e2} are the look-aheads limit of the base layer, the first enhancement layer and the second enhancement layer, respectively. In the example shown in Figure 4 we assume that the GOP size is 8 and there are 3 layers: base layer, E0, and E1 or layer 0, 1, and 2 respectively. The LASA algorithm starts scheduling at the NAL unit numbered 0 to the NAL unit numbered 7. The value of look-ahead for the base is therefore 8 that is $L_{base} = 7$. For layer E0 and E1, $L_{e1} = 5$ and $L_{e2} = 3$ respectively. DeltaL1 is the difference between layer 0 and layer

1 and deltaL2 is the difference between layer 2 and layer 1. Therefore, $deltaL_i$ is the difference between layer i and i−1. The pseudocode of the LASA scheduling algorithm is shown in Figure 4.10.

We can consider that frame based scheduling is when the parameters are assigned to be deltaL = 0, $L_{base} = 1$, $L_{e1} = 1$ and $L_{e2} = 1$, respectively and layer-base scheduling is when deltaL = 0, $L_{base} \geq 1$, $L_{e1} \geq 1$ and $L_{e2} \geq 1$ respectively. For LASA, the parameters DeltaL > 0, $L_{base} \geq 1$, $L_{e1} \geq 1$ and $L_{e2} \geq 1$, respectively.

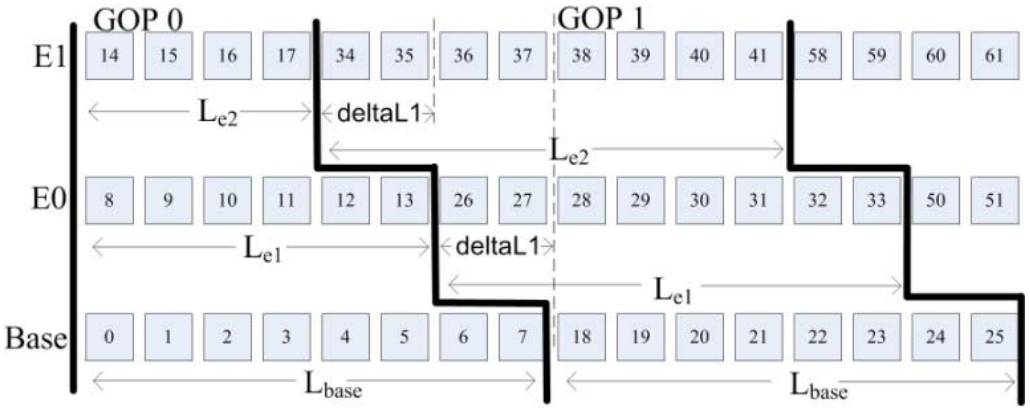

Figure 4.9 Look-ahead value is allocated to each layer unequally.

LASA Algorithm:

----------*Inintial Section*----------

- initialize LookAhead of Base layer (0^{th}) to
  MaxLookAhead
- initialize LookAhead of Enhancement layer i (i^{th}) to
  MaxLookAhead-$\sum_{x=1}^{x=i} DeltaL_x$,where i is top layer.
- initialize current upper bound of layer i^{th} to
  LookAhead of layer i^{th}
- initialize current NAL Unit index of layer i^{th} to 0

----------*Main Section*----------

While (NOT All Layers Reach End Of Frame Sequence)

For each layer i = 0^{th} to Top Layer

While (current NAL Unit index of layer i^{th} <=
current upper bound of layer i^{th}
AND
current NAL Unit index of layer i^{th} <= LastFrame
)

- send current NAL Unit of layer i^{th}

Figure 4.10. The pseudocode of LASA scheduling algorithm.

4.6 An Analysis of LASA Algorithm

This section is dedicated to the analysis of the LASA algorithm in which we show that the sending time for NAL units in the base layer, scheduled by LASA have sending times less or equal to layer-based scheduling.

We define the following terms:

$t^{s}_{lay_{[g,l,i]}}$ is the time when layer-based scheduling starts sending a NAL unit g, l, i where

g is a group of pictures number, $g \in I, g \geq 0$

l is the layer number, $l \in I, 0 \leq l \leq l_{max}$

i is the sending order of frames, $i \in I, 0 \leq i \leq gs$.

$t^{s}_{lh_{[g,l,i]}}$ is the time when LASA starts sending the first bit of the NAL unit g, l, i

$t^{p}_{[g,l,i]}$ is the time spent between the beginning of the first bit and the end of a NAL unit g, l, i

l_{max} is the number of layers, so $l_{max} \geq 0$

gs is GOP size, so $gs \geq 1$

Figure 11 shows the meaning of $t^{s}_{[g,l,i]}$ and $t^{p}_{[g,l,i]}$.

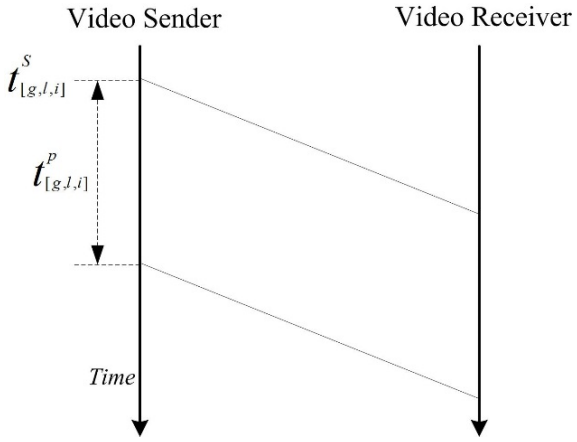

Figure .4.11. Definition of the time to start sending ($t^{s}_{[g,l,i]}$) and time spent for sending each NAL unit ($t^{p}_{[g,l,i]}$) used in this section.

The time to start sending any base layer NAL unit is given in Equation (4.2).

$$t^{s}_{lay_{[x,0,z]}} = \sum_{g=0}^{x-1} \sum_{l=0}^{l_{\max}} \sum_{i=0}^{gs-1} t^{p}_{[g,l,i]} + \sum_{i=0}^{z-1} t^{p}_{[x,0,i]} \quad (4.2)$$

where $x, z \in I \; and \; x \geq 1, z \geq 1$.

$$t^{s}_{lh_{[x,0,z]}} = t^{s}_{lay_{[x,0,z]}} - t^{p}_{skip} \quad (4.3)$$

where t^{p}_{skip} is the time spent for some NAL units in layer-based scheduling but is not used by LASA scheduling. $t^{p}_{skip} > 0$ when DeltaL1 or DeltaL2 > 0 as shown in Figure 6b,c.

Figure 4.12. The values of t^{p}_{skip} when

(**a**) DeltaL1 = DeltaL2 = 0 and $t^{p}_{skip} = 0$;

(**b**) DeltaL1 = DeltaL2 = 1 and $t^{p}_{skip} = t^{p}_{29} + t^{p}_{37} + t^{p}_{38}$;

(**c**) DeltaL1 = DeltaL2 = 2 and $t^{p}_{skip} = t^{p}_{26} + t^{p}_{27} + t^{p}_{34} + t^{p}_{35} + t^{p}_{36} + t^{p}_{37}$.

In Figure 4.12 (b), the time spent sending the NAL unit numbers 29, 37 and 38 is t^{p}_{skip} of Equation (4.3) when DeltaL1 = DeltaL2 = 1. So, if we use LASA, the NAL unit marked by B,

which would otherwise be numbered 21, will be sent earlier than scheduled by the layer-based approach, as shown in Equation (4.4).

$$t^{s}_{lh_B} < t^{s}_{lay_B} \tag{4.4}$$

where:

$t^{s}_{lh_B}$ is the time when LASA starts sending NAL unit B.

$t^{s}_{lay_B}$ is the time when the layer-based scheduling algorithm starts sending NAL unit B.

$t^{p}_{29}\ t^{p}_{37}\ and\ t^{p}_{38}$ are the times spent for sending NAL units 29, 37 and 38, respectively.

In Figure 4.12 (c), the time spent sending the NAL units 26, 27, 34, 35, 36 and 37 is t^{p}_{skip} when DeltaL1 = DeltaL = 2. Therefore, if the NAL unit marked by C is scheduled by LASA, it will be sent earlier than by layer-based scheduling as shown in Equation (4.5).

$$t^{s}_{lh_C} < t^{s}_{lay_C} \tag{4.5}$$

With reference to Equation (4.6), due to $t^{p}_{skip} > 0$, every NAL unit of the base layer which is scheduled by LASA will be sent earlier than the corresponding layer-based scheduling algorithm.

$$t^{s}_{lh_{[x,0,z]}} < t^{s}_{lay_{[x,0,z]}} \tag{4.6}$$

In the case of the first GOP where $t^{p}_{skip} = 0$, LASA has not skipped any NAL units. Therefore, in this case, we have

$$t^{s}_{lh_{[x,0,z]}} = t^{s}_{lay_{[x,0,z]}} \tag{4.7}$$

Referring to Equations (4.6) and (4.7), we can conclude that the starting time of any base layer NAL unit transmitted by LASA is less than or equal to that of the layer-based scheduling algorithm. The "sooner the better" approach for scheduling the base layer adopted by LASA is the reason the algorithm is able to provide more jitter protection to the base layer. In unpredictable traffic conditions, it is better if base layer NALs have as much time as possible to allow for the unpredictable delay.

Therefore, the result of comparing LASA scheduling and layer scheduling algorithms with respect to sending time is shown in Equation (4.8).

$$t^{s}_{lh_{[x,0,z]}} \leq t^{s}_{lay_{[x,0,z]}} \tag{4.8}$$

Next, layer-based and frame-based scheduling algorithms are analyzed.

$$t^{s}_{frm_{[x,0,z]}} = \sum_{g=0}^{x-1}\sum_{i=0}^{gs-1}\sum_{l=0}^{l_{\max}} t^{p}_{[g,l,i]} + \sum_{i=0}^{z-1} t^{p}_{[x,0,i]} + \sum_{i=0}^{z-1} t^{p}_{[x,1,i]} + \sum_{i=0}^{z-1} t^{p}_{[x,2,i]} \tag{4.9}$$

With reference to Equations (4.2) and (4.9), note that, the first and the second terms of Equations (4.2) are (4.9). Therefore

$$t^{s}_{frm_{[x,0,z]}} = t^{s}_{lay_{[x,0,z]}} + \sum_{i=0}^{z-1} t^{p}_{[x,1,i]} + \sum_{i=0}^{z-1} t^{p}_{[x,2,i]} \tag{4.10}$$

Because of $\sum_{i=0}^{z-1} t^{p}_{[x,1,i]} \geq 0$ and $\sum_{i=0}^{z-1} t^{p}_{[x,2,i]} \geq 0$, so we obtain

$$t^{s}_{frm_{[x,0,z]}} \geq t^{s}_{lay_{[x,0,z]}} \tag{4.11}$$

From Equations (4.8) and (4.11), we conclude that the base layer NAL unit will receive greater benefit using LASA scheduling than layer-based and frame-based scheduling with respect to the start time of transmission. The order of the starting times for the three different scheduling algorithms is shown in Equation (4.12).

$$t^{s}_{lh_{[x,0,z]}} \leq t^{s}_{lay_{[x,0,z]}} \leq t^{s}_{frm_{[x,0,z]}} \quad (4.12)$$

In the next section, the performance of the LASA scheduling algorithm will be evaluated through simulation.

4.7 Simulation

This section will be dedicated to the simulation of the LASA algorithm. The simulation is conducted in rapidly fluctuating bandwidth using network simulator (NS3).

4.7.1 Simulation Setup

Figure 4.13 depicts the dumbbell topology as constructed for the simulation. Two pairs of end nodes are connected together and share the bottleneck link. All link capacity was set to 1 Mb/s and 20 ms for transmission delay. In order to generate rapidly fluctuating bandwidth available, ON/OFF UDP flow with exponential random variable has been used as background traffic by setting the mean of ON time period varied from 0.2 to 0.4s and mean of OFF time period was constant at 1s and the mean UDP packet is set to 1000 bytes.

We concatenated five public frame sequences which are comprised of mobile, soccer, foreman, harbor and city. All videos are 30 fps, 300 frames and CIF (352x288) resolution. This concatenated video was encoded and decoded by JSVM reference software [91] and was

injected to NS3 simulator. The encoded video is composed of a base layer and two quality layers. We used GOP of 16 frames. In each simulation ran, we injected the bit-stream of the concatenated video to the network simulator by setting 600s of simulation time for each run. We also repeated our simulation with different seeds of random variables in order to vary the background traffic pattern.

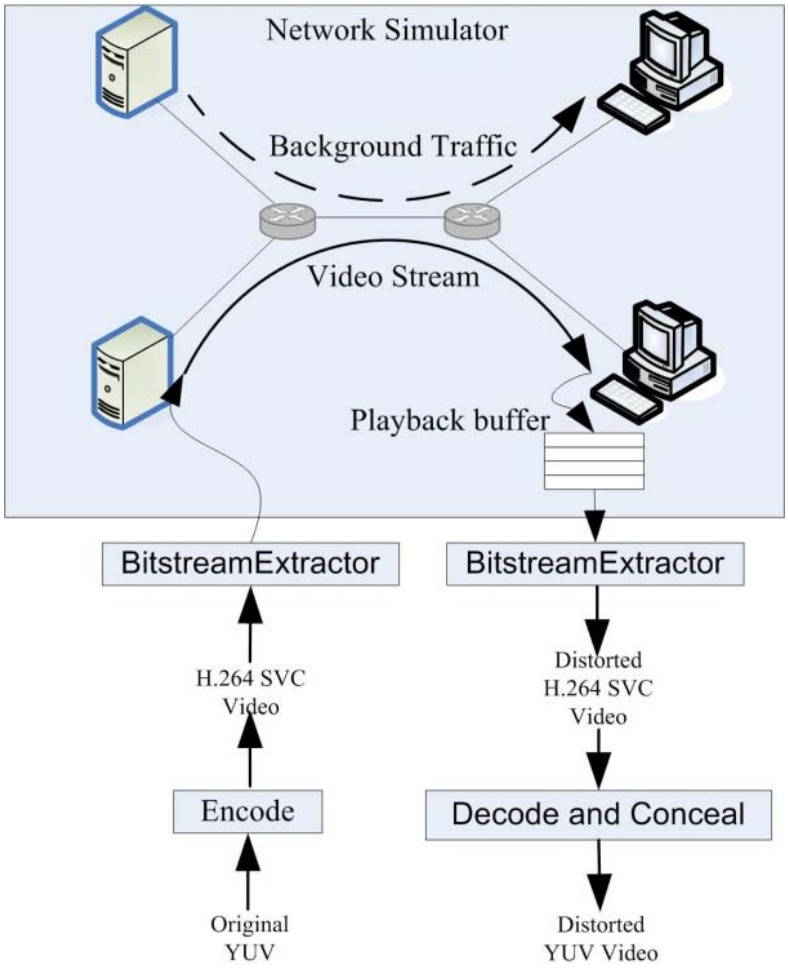

Figure 4.13 Network topology and input/output video stream used in this simulation.

All three scheduling algorithms (LASA, layer-based, and frame-based) were implemented and compared. In the same way as other streaming video research, the buffer has

also been added as part of the simulation to collect NAL units. When pre-buffering threshold is reached, the receiver will start decoding and playout [96].

In the playback buffer management module, it also stamps the receiving time of each NAL unit. If some NAL units arrive at the receiver later than their deadlines, those NAL units will be marked as deadline violations which will not be passed to the decoder. We collected the receiving time of each NAL unit to analyze the number of frames that miss their deadlines.

All undecodable frames either the missed deadline frames, lost frames or erroneous frames will be concealed with the previous frame in order to maintain the number of frames in the video sequence. In line with other researchers [106] we compare Y PSNR between the original YUV and distorted YUV as shown in Figure 4.13.

4.7.2 Performance Evaluation

In this experiment, we compare the performance of the three algorithms by concentrating on the continuity and the average Y PSNR. The result of LASA scheduling reveals that the number of frames that miss the deadline and the number of concealed frames is lower than the layer-based and frame-based scheduling algorithms. LASA scheduling with deltaL equal to 2, 4 or 8 is able to improve results but for deltaL equal to 16 the average Y-PSNR is very similar to layer-based scheduling, but higher than the frame-based method.

The essential parameters are varied. Firstly, the ON time period of background traffic is varied between 0.0 to 0.4 s to produce jitter. Secondly, we vary the pre-buffering threshold of the playback buffer. The larger is the playback buffer the better it is in terms of its ability to resist jitter. We, therefore, vary the period of time before beginning playback from 400 to 600 ms.

In Figures 4.14 - 4.16, the percentages of missed deadlines among the three algorithms are observed.

In Figure 4.14, the playback buffer threshold is set to 600 ms, the percentage of missed deadline frames for LASA scheduling is lower than layer-based scheduling. LASA scheduling is able to reduce the number of frames missing the deadline considerably. For the case of LASA with deltaL equal to 16, when high burst duration (0.2 to 0.4) is injected, the percentage of missed deadline frames is 0.84% lower than layer-based scheduling on average.

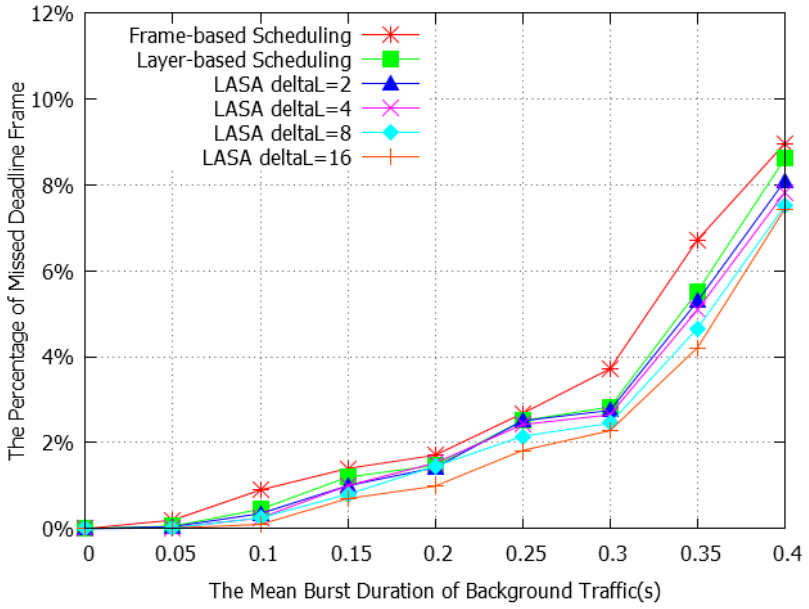

Figure 4.14. The number of frames with deadline violation when the threshold is 600 ms.

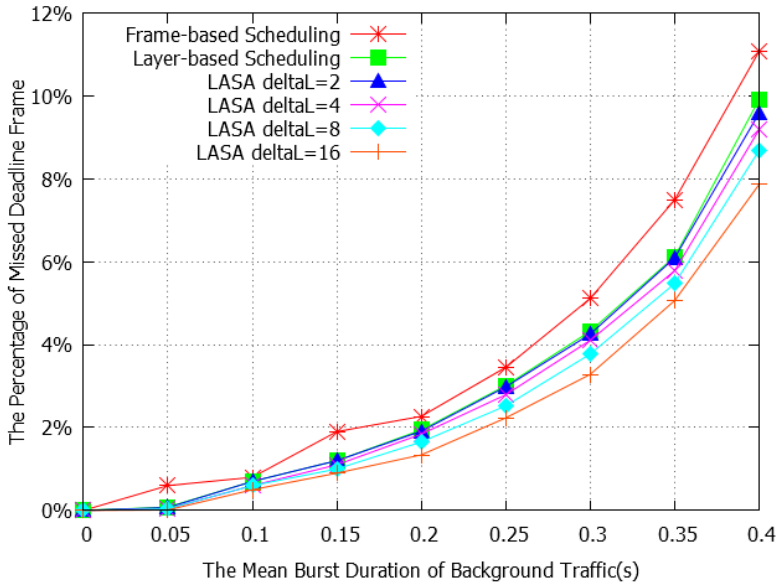

Figure 4.15. The number of frames with deadline violation when the threshold is 500 ms.

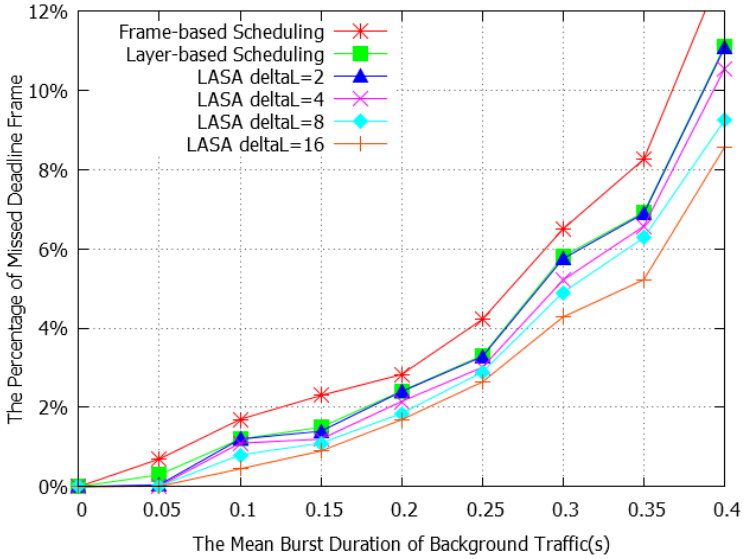

Figure 4.16. The number of frames with deadline violation when the threshold is 400 ms.

To increase the influence of jitter we reduced the play out buffer from 600 to 500 ms and 400 ms as shown in Figures 9 and 10. As summarized in Table I, the largest improvement provided by LASA is when deltaL = 16; around 1.43%. When deltaL are 2, 4 or 8, the percentage of missed deadlines are still lower than the layer-based approach. The negative

values seen in Table 1 are representative of when LASA is providing improved results over layer-based scheduling.

Table 4.1. Comparison of the percentage of missed deadline frames between layer-based and the LASA scheduling algorithm when burst duration is varied from 0.2 to 0.4 s.

Threshold	deltaL = 2	deltaL = 4	deltaL = 8	deltaL = 16
400 ms	−0.03	−0.42	−0.88	−1.43
500 ms	−0.01	−0.32	−0.64	−1.11
600 ms	−0.18	−0.29	−0.54	−0.84
Average	−0.07	−0.34	−0.69	−1.13

LASA scheduling has the highest performance of the three scheduling algorithms with respect to the number of frames that have arrived at the receiver after the deadline for playback. The results for buffer thresholds of 600, 500, and 400 ms as shown in Figures 4.14 - 4.16, respectively. These results agree with the mathematical analysis in Equation (4.12) and our assumption that if we send important NAL units earlier, there is more time to overcome unpredictable delays, and the probability of missing the deadline will be reduced accordingly.

As some base layer NAL units may miss their decoding deadline, or being lost or are in errors, they cause some descendant NAL units not to be decoded even if they arrive at the receiver before the deadline. All undecodable frames will be concealed with their previous frames. The percentages of the undecodable frames are shown in Figures 4.17 - 4.20. From the user's perspective, if the number of frozen frames increases, the continuity of the video stream decreases.

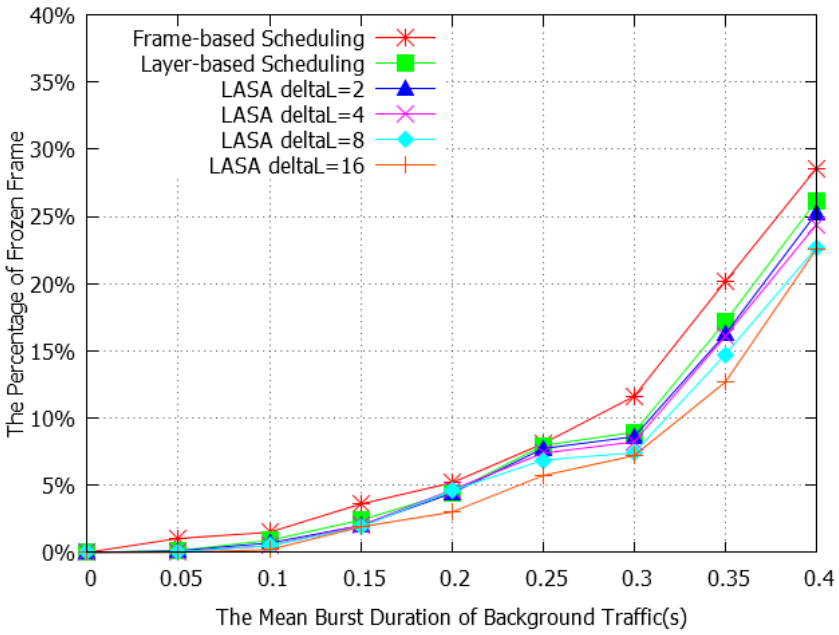

Figure 4.17. The number of frozen frames when threshold is 600 ms.

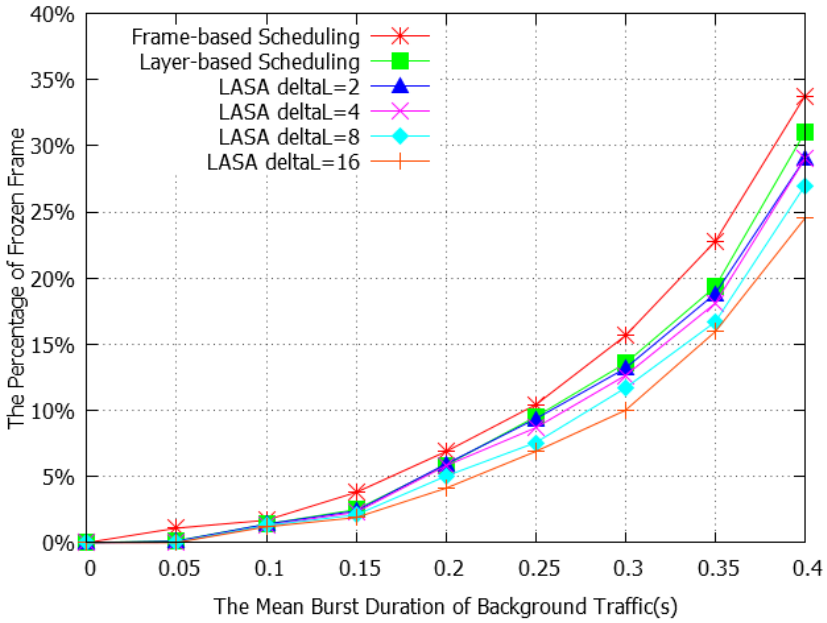

Figure 4.18. The number of frozen frames when threshold is 500 ms.

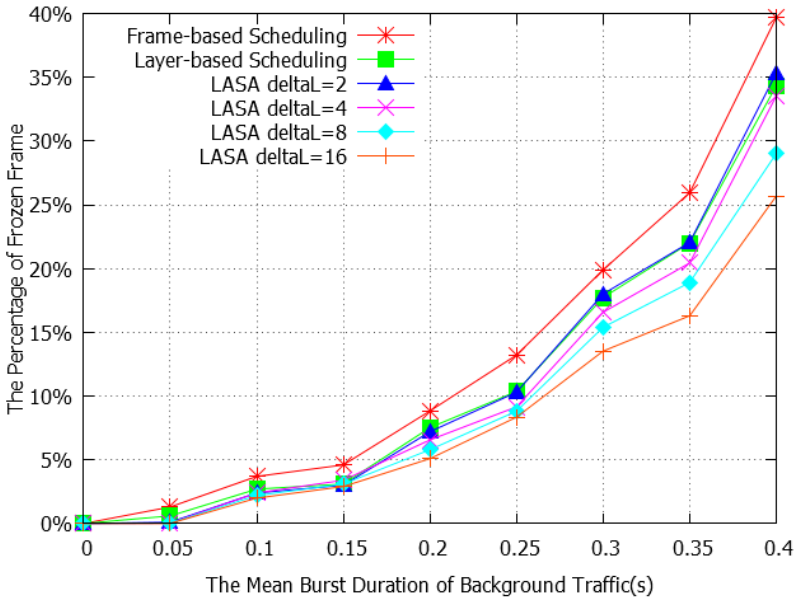

Figure 4.19. The number of frozen frames when threshold is 400 ms.

In Figure 4.17, the percentage of frozen frames for LASA scheduling is lower than layer-based and frame-based scheduling. For example, on high burst duration (0.2–0.4 s) LASA with deltaL = 16 is lower than layer-based scheduling by 2.71%. For thresholds of 500 ms, and 400 ms as shown in Figures 12 and 13, the percentages of frozen frames for LASA are lower than layer-based by 3.56% and 4.60% respectively. In Table 4.2, we compare the percentage of frozen frames between layer-based and LASA. The negative value corresponds to when LASA is better than layer-based in terms of frozen frames.

Table 4.2. Comparison of the percentage of frozen frames between layer-based and the LASA scheduling algorithm when burst duration is varied from 0.2 to 0.4s.

Threshold	deltaL = 2	deltaL = 4	deltaL = 8	deltaL = 16
400 ms	0.18	−1.14	−2.81	−4.60
500 ms	−0.59	−1.01	−2.27	−3.56
600 ms	−0.50	−0.80	−1.69	−2.71
Average	−0.30	−0.98	−2.25	−3.63

LASA scheduling is designed to give more protection to the base layer by sacrificing enhancement layer protection. However, since the loss of base layer data is more damaging than the loss of enhancement data, the overall video quality may not be degraded that much. Moreover, in the LASA algorithm even if we lose video quality in some frames but we gain video continuity by reducing the percentage of frozen frames compared to other rival algorithms. Hence the overall video quality is improved.

Figures 4.20 - 4.22 show that LASA scheduling with deltaL equal to 8 either for 400 ms, 500 ms or 600 ms thresholds is able to improve the Y PSNR significantly by 0.92 dB, 0.96 dB and 0.69 dB, respectively. For LASA scheduling with deltaL equals to 4, Y PSNR is improved by 0.44 dB, 0.43 dB and 0.37 dB. However, in the case of deltaL = 2 and deltaL = 16 the average Y PSNR is similar to layer-based scheduling. In Table 4.3, we summarize the average Y PSNR of LASA and layer-based scheduling. It should be noticed that, both LASA and layer-based scheduling are better than frame-based.

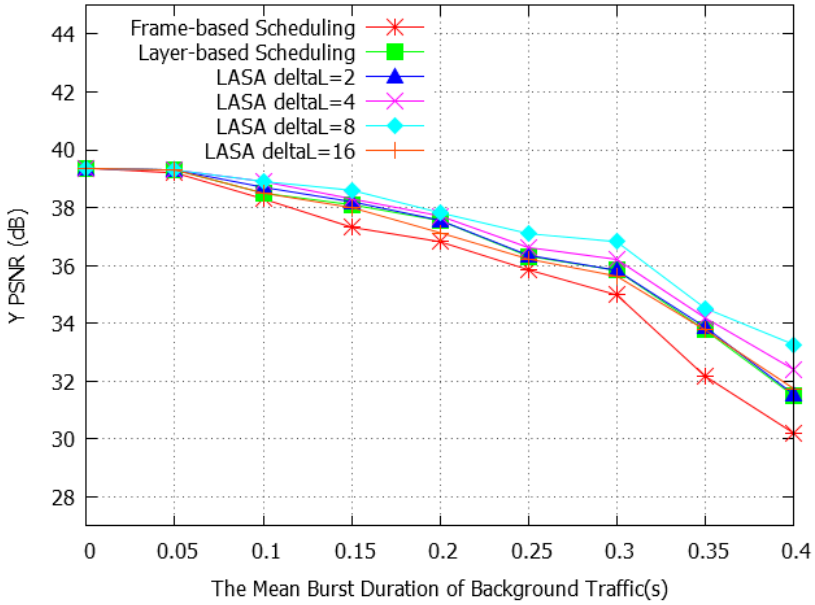

Figure 4.20. The average Y Peek Signal-to-Noise Ratio (PSNR) when the threshold of the playback buffer is 400 ms.

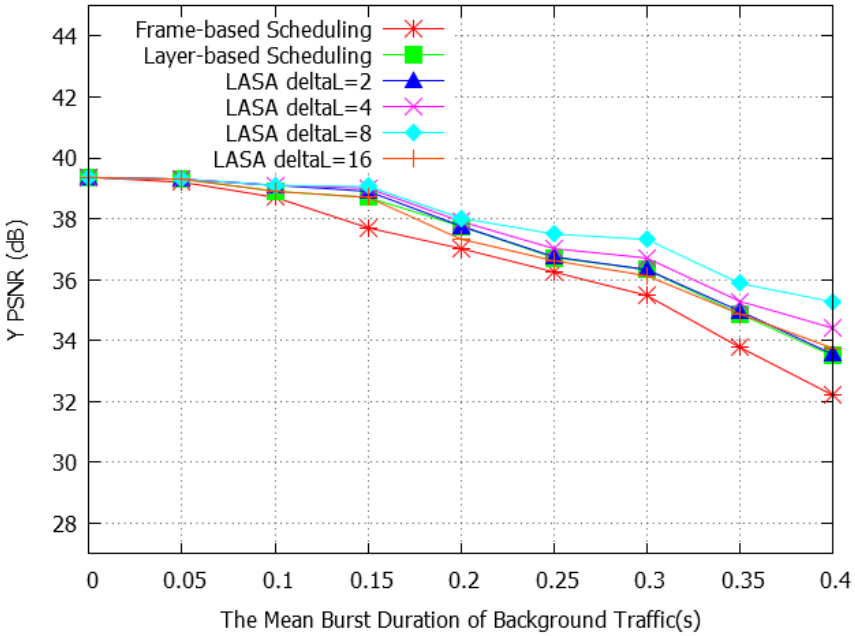

Figure 4.21. The average Y PSNR when the threshold of the playback buffer is 500 ms.

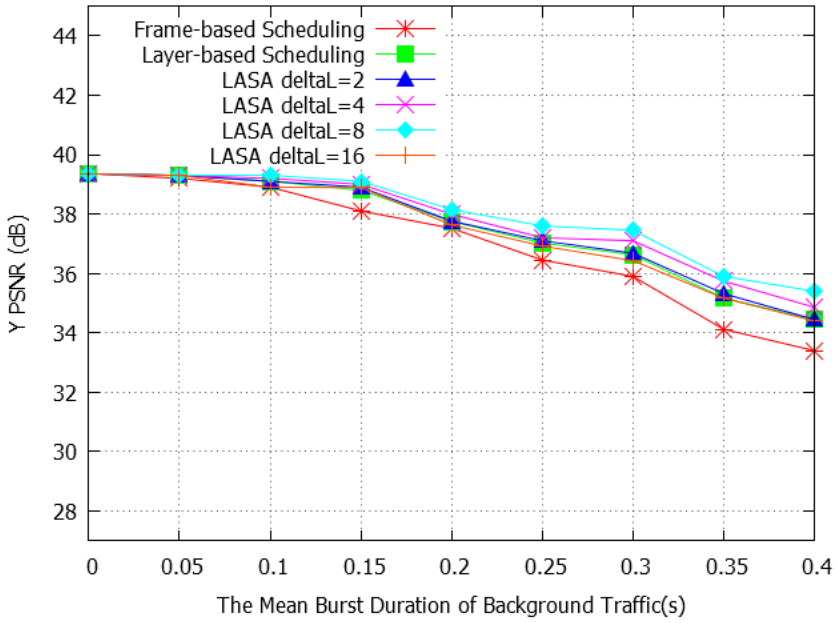

Figure 4.22. The average Y PSNR when the threshold of the playback buffer is 600 ms.

Table 4.3. Comparison of Average Y PSNR, in dB, between Layer-based and LASA scheduling algorithms when burst duration is varied from 0.2 to 0.4s.

Threshold	deltaL = 2	deltaL = 4	deltaL = 8	deltaL = 16
400 ms	0.04	0.44	0.92	−0.09
500 ms	0.04	0.43	0.96	−0.09
600 ms	0.06	0.37	0.69	−0.09
Average	0.05	0.41	0.86	−0.09

In Table 4.3, negative values represent when LASA has a lower PSNR than layer-based scheduling whilst a positive value represents higher. The difference of the average Y PSNR of LASA and layer-based scheduling vary between -0.09 dB to 0.96 dB and it is only when deltaL is equal to 16 that a negative value is obtained.

4.8 Chapter Summary

The main contribution of the LASA scheduling algorithm, presented in this chapter, is improved continuity of a video stream in a jitter-prone environment by providing an unequal look-ahead value to each layer of a scalable video. It is able to maintain or improve the average Y PSNR compared to layer-based and frame-based scheduling algorithms. In addition to the PSNR improvement of LASA by up to 4 dB, the continuity of the video is also improved. This quality measurement can be more significant subjectively, as at high video qualities, 4 dB improvement may not be appreciated, but continuity is notable.

By exploiting the flexibility of the scalable video architecture in which a video is composed of one or more layers of unequal importance, improvements in dropped frames and

PSNR are demonstrated. With predefined unequal look-ahead values, the base layer scheduled by LASA provides improved safety to the base layer and means the enhancement NAL units do not get discarded.

In addition, the novel approach for jitter resistance of LASA can be integrated with standard playback buffers to improve performance further, if higher end-to-end delay is permissible.

Chapter 5
Continuity-Aware Scalable Video Streaming Using an Adaptive Look Ahead Scheduling Algorithm

5.1 Introduction

The increase in user demand in requests and dissemination of visual content on packet networks has brought about improvement in various video-related business services and standards. Consequently, the rivalry between video service providers has resulted in various video streaming technologies being propelled into the business sector. One of the main problems that these technologies aim to solve is to create a video streaming technology that can combat the congestion of the packet networks.

In spite of the fact that today's overall Internet bandwidth has increased significantly, the actual bandwidth availability that is allocated to each flow of a video streaming session still fluctuates. Therefore when the bandwidth availability is deprived, it causes jitter or variation in inter-arrival time between the two successive video portions at the receiver, which then leads to an interruption of the video streaming session [26, 27, 31, 85]. In the jitter-prone network, the number of video frames to reach the receiver after the decoding deadline increases, and therefore a higher percentage of frames will not be decoded. Therefore, the average video quality is degraded [26, 31, 88]. In this chapter, we present an adaptive look ahead scheduling algorithm that can resist the jitter by adapting our previous models in chapter 3 and chapter 4.

In this chapter, we address the problem of discontinuity of video caused by the late arrival of NAL units. By exploiting the advantages of the scalable video by providing an

unequal look ahead to the scalable video layers, we introduce a novel scheduling algorithm that provides automatic adjustment of unequal jitter protection to a scalable video stream called the Adaptive Look Ahead Scheduling Algorithm (ALASA).

The video stream could be smoothed by using playback buffer management but at the expense of startup delay time. Our scheduling algorithm can collaborate with a buffer mechanism and provide further improved performance. The overview architecture of the adaptive scheduling algorithm is demonstrated in Figure 5.1.

The rest of this chapter is structured as follows. Section 5.2 presents related work. The Look Ahead Scheduling Algorithm will be presented in Section 5.3. The simulation and its results are described in Section 5.5, and Section 5.6 is the conclusion.

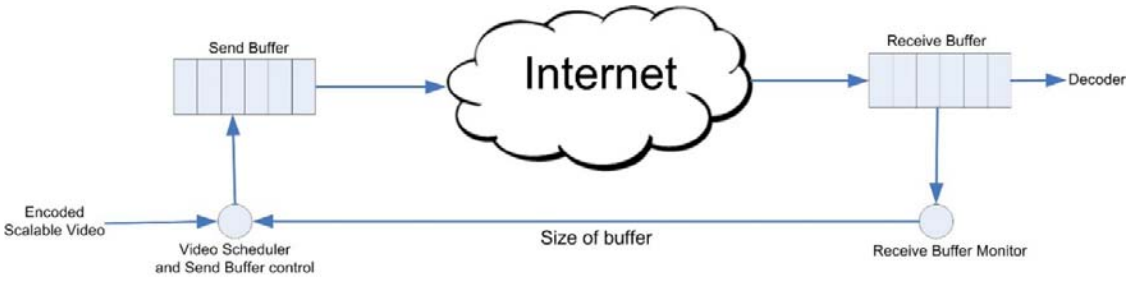

Figure 5.1 Overview Architecture of Adaptive Look Ahead Scheduling Algorithm

5.2 Related Works

Researchers in [107] developed an adaptive unequal protection strategy for packet loss in the scalable video stream using Luby transform coding. This adaptive algorithm decreases the number of quality layers of a group of pictures (GOP) in Multimedia Intelligent Surveillance (MIS) to protect more localised MIS in limited channel resources when a high packet loss rate is detected. Otherwise, the number of quality layers in MIS is increased to improve video quality in a lower packet loss rate condition.

The study in [48] proposed a delay-sensitive scheduling algorithm. The scheduling algorithm assigns higher priority to high delay packets by using delay information of queues and the number of lost packets in the different queues as parameters. As the priority driven packet scheduling has a direct and significant impact on overall delay bound behavior of the packets, the research in [51] proposed delay bound constraints and priority-based packet scheduling for video delivery, which predicts the probability of packets that would be dropped because of the violation of the delay limit. The research in [108] proposed and evaluated the time-constraint optimized packet scheduling algorithm, which minimized the sum of expected distortion under the condition of delay constraints. This scheduling algorithm uses a Markov chain model to predict the network condition and then drop some packets in order to release network resources to the rest of the packets. The research in [20] proposed a fairness-aware smooth rate adaptation algorithm for DASH under the scenario that multiple clients are sharing the network channel. This algorithm uses probe-based bandwidth to estimate the fair-share bandwidth and then adjust the bit rate of each video stream. To improve the quality of scalable video stream, the study in [109] proposed adaptive quality optimization algorithms to maximize clients' QoE and smoothly adapt service quality according to resource availability.

5.3 Adaptive Look Ahead Scheduling Algorithm

The Look Ahead Scheduling Algorithm was presented and investigated in chapter 4. However, in chapter 4 we investigated its performance only in terms of look ahead parameters (deltaL) that are constant. In this chapter, we extend the look ahead scheduling algorithm by concerning the adapting of the look ahead parameters-deltaL of each video layer to correspond with the bandwidth availability. The size of the play-out buffer returned by the receiver is considered to be the primary parameter to adjust the deltaL parameters.

The adaptive look ahead scheduling algorithm is expected to reduce the interruption level of video flow appearing to the viewer. The interruption level will be indicated by the percentage of the freeze frames. Furthermore, the overall quality of the video is investigated through the average Y-PSNR value.

5.3.1 Overview of Adaptive Look Ahead Scheduling Algorithm

The adaptive look ahead scheduling algorithm needs many parameters to maintain the continuity of a scalable video stream. In this section, we introduce the most important parameter called the "boundary limit", which is varied over time according to the level of congestion detected.

Figure 5.3 shows the comparison of the look ahead scheduling algorithm and the traditional scheduling, layer-based scheduling algorithm. The number order demonstrates the sequence of the sending pattern. Each square with a number labelled presents a scalable video NAL unit.

Let us consider the look ahead scheduling algorithm in Figure 5.3 (b); the boundary of each video layer is not equal. The base layer's boundary limit is far more than the boundary limit of layer E0 and the boundary limit of layer E0 is far more than layer E1. In the case of the adaptive look ahead scheduling algorithm, the boundary limit of each layer will be automatically adjusted when the size of the play-out buffer changes.

Indeed, we might consider that the layer-based scheduling algorithm is a static window while the adaptive look ahead scheduling algorithm is a variable window size. In this case, the idea of the look ahead scheduling algorithm is similar to the idea of a variable window size [85, 86, 110, 111] that has been used in the reliable transport protocol, such as TCP.

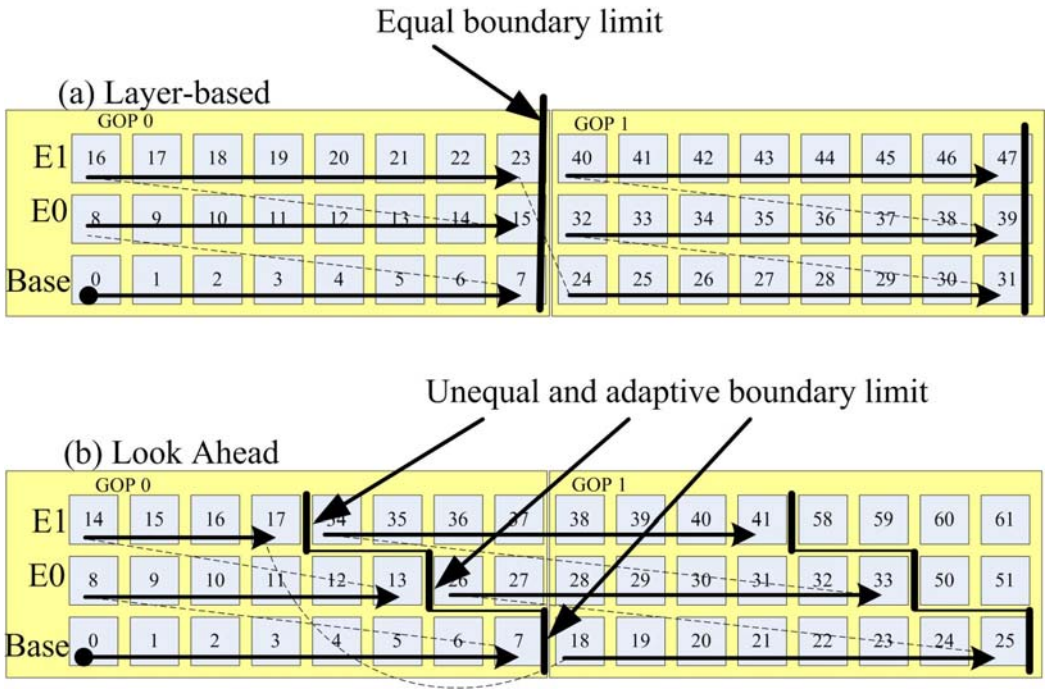

Figure 5.3 Comparison of a Layer-based and Look Ahead Scheduling Algorithm.

The boundary limit of each layer is defined by the deltaL parameter. It is an essential parameter which defines the difference between any layer i and its underlying layer. For example in Figure 5.4, when deltaL1 is set to 2, it determines the boundary limit of the base layer when it is far more than E0 2 units. For deltaL2, the E0 layer boundary limit of E1 is far more than E2 2 units when it is set to 2. These parameters lead to the rise of the the jitter protection level given to the underlying layer. The more $deltaL_i$ is increased, the more tolerant the variation of the end-to-end delay is.

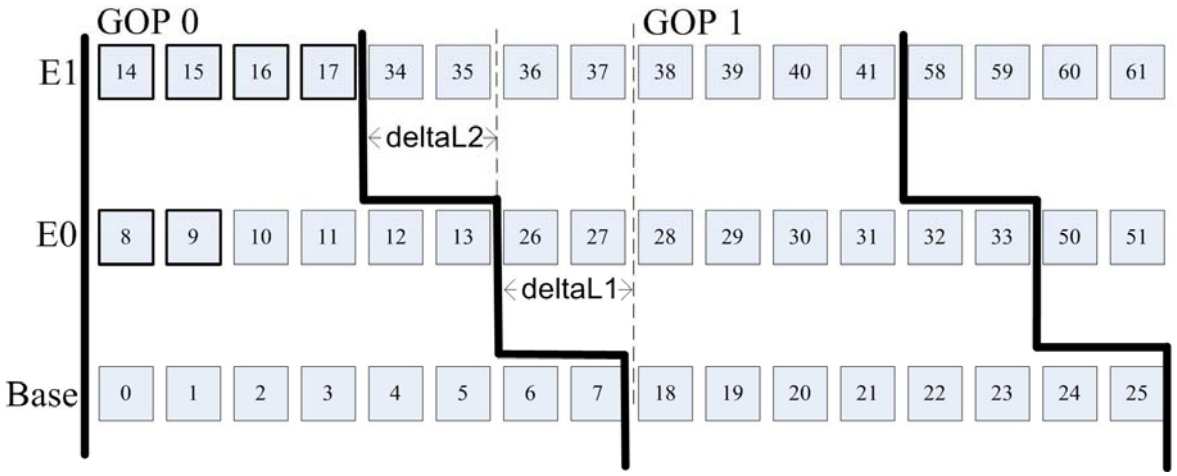

Figure 5.4 The sending pattern and parameters of the look ahead scheduling algorithm.

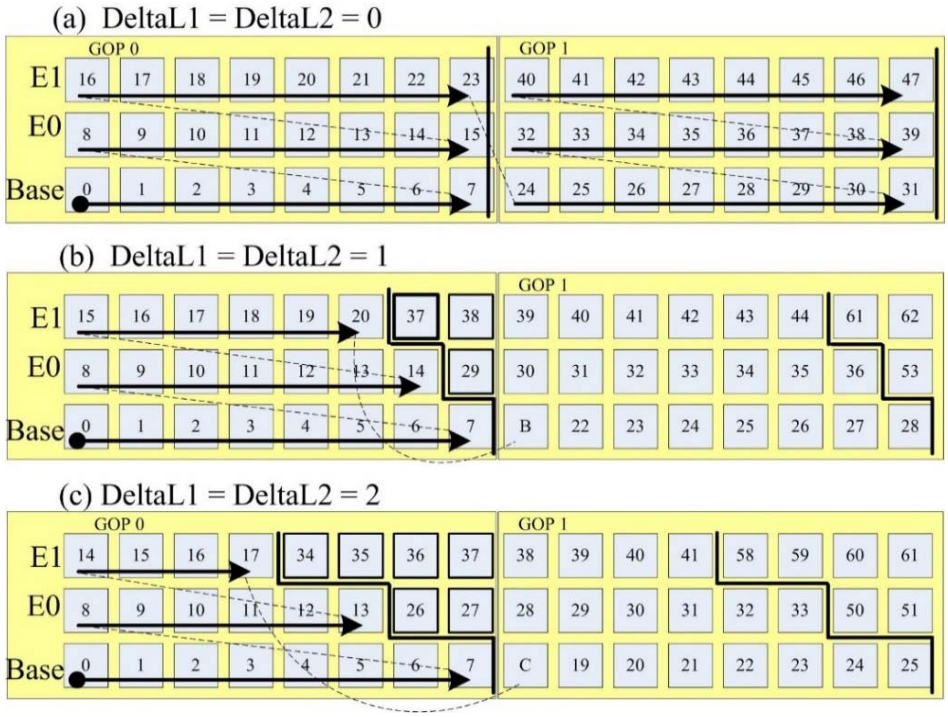

Figure 5.5 DeltaLi when it is set to a different value,

(a) rectangular sending window, LASA when the DeltaL1 = DeltaL2 = 0

(b) non-rectangular sending window, LASA when DeltaL1 = DeltaL2 = 1

(c) non-rectangular sending window, LASA when DeltaL1=DeltaL2 = 2

In principle, when the adaptive look ahead scheduling algorithm attempts to give more protection level to the more important layer, the adaptive look ahead scheduling algorithm will provide a higher protection level to the base layer than the layer E0. If we compare the lookahead scheduling algorithm with traditional layer-based scheduling, the drawback of the look ahead scheduling algorithm is that it will put more risk to E0 and E1. Therefore, layer E0 and E1 tend to reach the receiver after the decoded time.

As shown in Figure 5.5 (b), when deltaL1 and deltaL2 are set to be 1, the bold square presents that the probability of layer E0 and layer E1 arriving at the receiver after the decoded time is higher than the case of when deltaL1 and deltaL2 are set to 0. According to the fact that the look ahead scheduling algorithm rearranges the transmission order of all video portions, some scalable video NAL units will get the benefit, but some scalable video NAL units have to be sacrificed. The look ahead scheduling algorithm guarantees that some scalable video NAL units that will get the benefit will always be the base layer.

For Figure 5.5 (c), if the deltaL1 and deltaL2 are set to be 2, the number of sacrificed scalable video NAL units is increased compared to the Figure 5.5 (b). Therefore, the base layer will get a higher jitter protection level because there are more NAL units in layer E0 and layer E1 that are considered to be the sacrificed portion. However, a decrease of the percentage of freeze frame will be noticed because the look ahead scheduling algorithm guarantees to reduce the base layer NAL units arriving receiver after the deadline.

5.3.2 Finite State Machine of Look Ahead Scheduling Algorithm

In this section, we will describe how the look ahead scheduling algorithm changes parameter detaL by using a finite state machine to represent the state of the algorithm.

The adaptive look ahead scheduling algorithm utilizes the amount of base layer's NAL units stored in the receiver's play-out buffer (denoted by parameter v in Figure 5.6) to be the parameter to define the state of the adaptive look ahead scheduling algorithm. The parameter v will be returned from receiver to video server.

Figure 5.6 shows the finite state machine of the adaptive look ahead scheduling algorithm. In the state S, the video server disseminates the scalable video NAL units to the receiver until the level of base layer's NAL units in the play-out buffer is greater than the upper threshold t_u and then it causes the finite state machine to change to state 0. In state 0, video viewer application starts consuming the NAL unit and thus decoding of the relevant NAL units now begins. In this state, if the incoming scalable video rate is more than the consumed rate, the amount of base layer stored in play-out buffer (denoted by parameter v in Figure 5.6) is decreased. The adaptive look ahead algorithm will stay in state 0 as long as the amount of base layer NAL units stored in play-out buffer is above the lower threshold (denote by parameter $t_l$in Figure 6.6). However, if the quantity of base layer's NAL unit is lower than the threshold t_l, look ahead scheduling algorithm will jump to state 1.

The adaptive look ahead scheduling algorithm is expected to tolerate the variation of an end to end delay. Therefore, the different characteristics of state 0 and state 1 are the tolerance level to jitter. To achieve this aim, the parameter deltaL of state 1 is greater than in state 0 to guarantee a higher continuity level of video, as described in section 5.3.1.

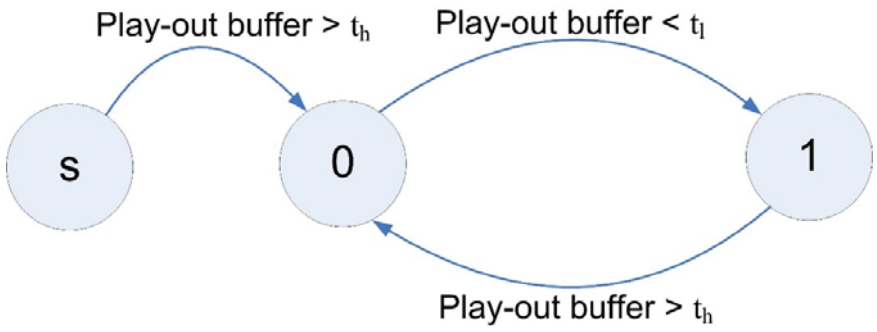

Figure 5.6 Finite state machine of look ahead scheduling algorithm

5.3.3 Adjusting the State of the Look Ahead Scheduling Algorithm

To adjust the state of the look ahead scheduling algorithm, we simplify the notion proposed in chapter 4 to estimate the congestion level. The parameters relating to the play-out buffer at the receiver is shown in Figure 5.7. We separate the area in the receiver's play-out buffer into three zones by using two parameters. The t_u and the t_l are the upper threshold and lower threshold, respectively. The two parameters play an important role in the adaptive look ahead scheduling algorithm as introduced in section 5.3.2 so far.

The receiver's play-out buffer Zone 0 represents the safest area. If the highest timestamp NAL unit indicated by parameter t_h is stored in this area, it means that there are plenty of base layer NAL units stored and the congestion level is considered to be low level. Next, if the highest timestamp NAL unit is stored in zone 1, the state of the adaptive look ahead scheduling algorithm will not change. Finally, if the highest timestamp NAL unit is stored in zone 3, it indicates that the congestion level will increase so the adaptive look ahead algorithm needs to be changed to state 1 where the base layer's protection level is higher than state 0.

The description of each parameter relating to the base layer's buffer is presented in Table 5.1.

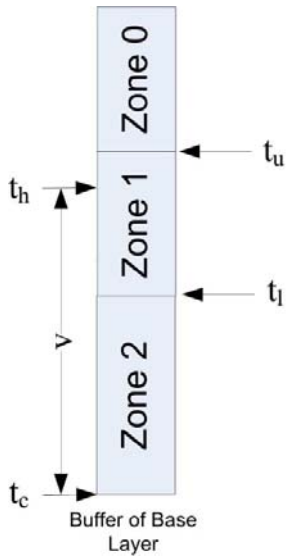

Figure 5.7 Video Buffer Time of Base Layer

Table 5.1 Parameters that represent the state of the base layer's buffer

Parameter	Description
v	denotes the video buffer time of the base layer's buffer
t_h	denotes the highest time stamp of the base layer's buffer
t_c	denotes the timestamp of the current playback time
t_u	denotes the upper threshold
t_l	denotes the lower threshold

The video buffer time (v **)** is calculated using equation (5.1). The parameter v will be calculated by the receiver and returned to the server. Actually, in our implementation, the server

holds parameter t_u and t_l. After the server receives parameter v, it will make a decision as described in section 5.3.2 of the finite state machine of the adaptive look ahead scheduling algorithm

$$v = t_h - t_c \tag{5.1}$$

5.3.4 Example of the Boundary Limit Adaptation

When the adaptive look ahead scheduling algorithm changes state, the boundary limit of each layer is adapted as shown in Figure 5.8 and Figure 5.9. Figure 6.8 shows the algorithm when it is in state 0. After the server receives parameter v returned back by the receiver, the adaptive scheduling algorithm makes a decision depending on parameter v. In this case, the parameter deltaL is increased. An example of when the adaptive look ahead scheduling algorithm adapts the state from state 1 to 0 is shown in Figure 5.9. In this case, the parameter deltaL is decreased.

In the next section, we will introduce the pseudocode of the adaptive scheduling algorithm.

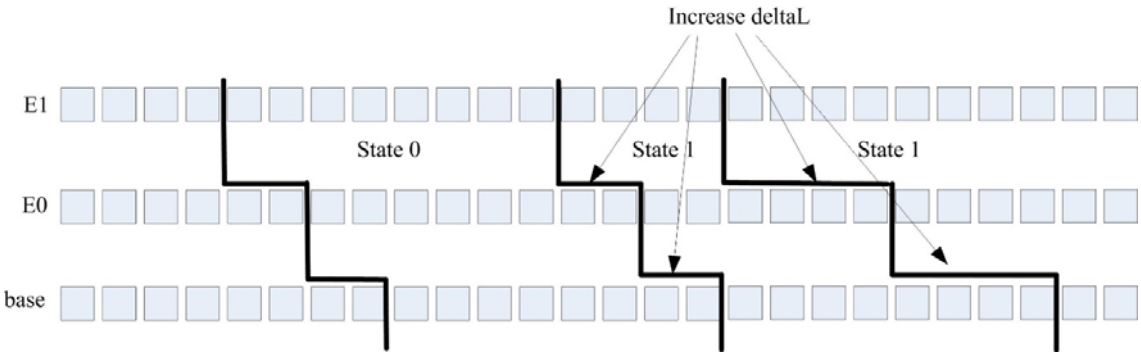

Figure 5.8 Adaptive look ahead scheduling algorithm changes from state 1 to 0

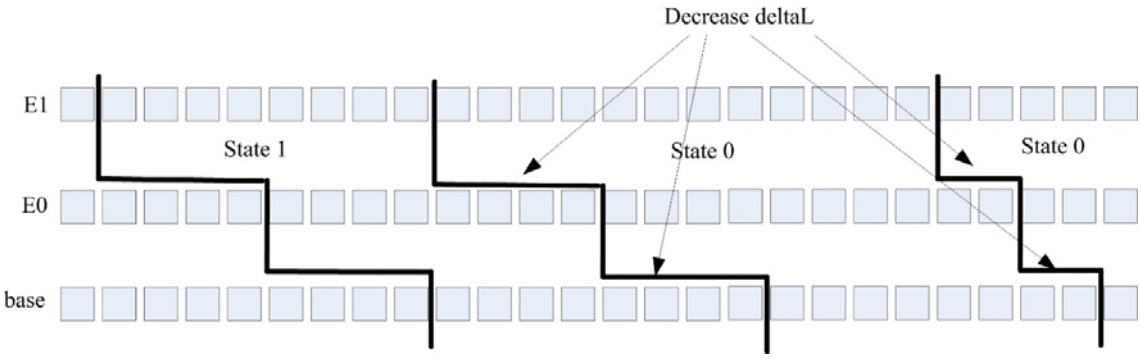

Figure 5.9 Adaptive look ahead scheduling algorithm changes from state 0 to 1

5.4 The Proposed Adaptive Look Ahead Scheduling Algorithm

In this section, we will describe the adaptive scheduling algorithm in detail. The initial section and main algorithm are shown in Figure 5.11 and Figure 5.12, respectively. The initial section is the same as that of the look ahead scheduling algorithm. It gives the initial value to the variable LookAhead of each layer.

The variable LookAhead is the amount of NAL units that will be transmitted in the next round. The role of the variable LookAhead is transformed to the diagram shown in Figure 5.10. For example, L_{base} is the variable LookAhead for a base layer. So, in Figure 5.10, the amount of the base layer's NAL units that will be transmitted in the first round are the NAL units labeled by numbers 0 to 7. Next, the adaptive look ahead scheduling will jump to the enhancement layer (E0) labeled by the number 8 and repeat sending E0's NAL units labeled by number 9, 10 and so on until the last NAL unit labeled number 13, which is the current boundary limit of layer E0.

To adjust the tolerance to the jitter level, the LookAhead has to be varied. Actually, the variable LookAhead defines the boundary limit described in section 5.3. The variable

LookAhead is used in the main section (Figure 5.12) at line 12. After finishing the while-loop (line 3 to 9), the variable LookAhead will be assigned to a new value using function 2 as shown in Figure 5.13.

The variable LookAhead will be assigned to a possible value according to the current state of the adaptive look ahead scheduling algorithm and the pre-defined value represented in Table 5.2. These pre-defined deltaL values represented in Table 5.2 will also be used in the simulation section (section 5.5).

One more feature added to the adaptive look ahead scheduling algorithm is checking the status of the ancestor's NAL unit, whether it has been sent or it has been dropped. If an ancestor's NAL unit was dropped, the current NAL unit would be dropped as well. For example, if an E0's NAL unit was dropped, the relating E1's NAL unit will also be dropped. This is function 1 in Figure 5.14. In addition, function 1 in Figure 5.14 will drop an NAL unit if current time plus the estimate end-to-end delay is greater than the play-out deadline.

The variable LookAhead is the most important variable reflecting the overall mechanism of the adaptive look ahead scheduling algorithm. In table 5.2, it is the pre-defined value that will be used in our simulation in the next section.

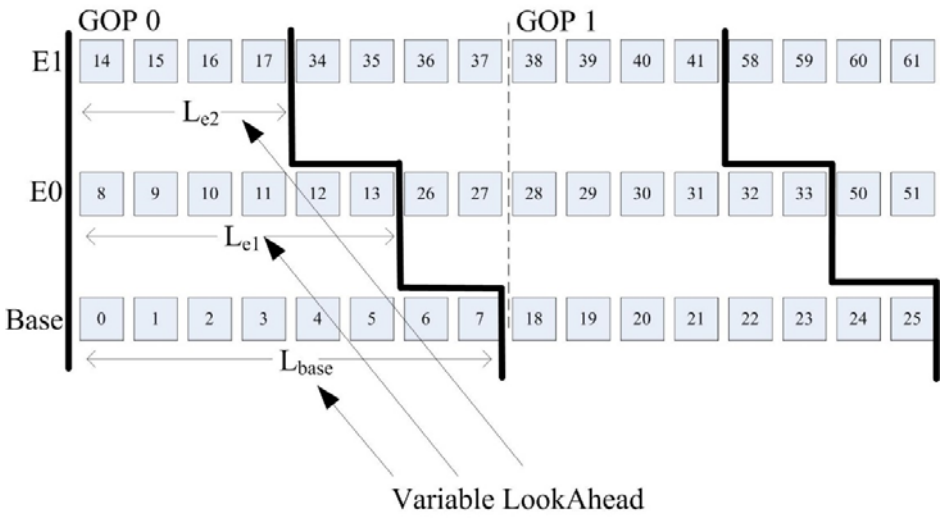

Figure 5.10 The usage of the variable LookAhead in the adaptive look ahead scheduling algorithm

----------*Initial Section*----------

1 initialize LookAhead of Base layer (0^{th}) to MaxLookAhead

2 initialize LookAhead of Enhancement layer i (i^{th}) to

MaxLookAhead-$\sum_{x=1}^{x=i} DeltaL_x$,where i is top layer.

3 initialize current upper bound of layer i^{th} to

LookAhead of layer i^{th}

Figure 5.11 The initial section of adaptive look ahead scheduling algorithm

```
Adaptive Look Ahead Scheduling Algorithm:
------------------------------Main Section-------------------------------
1 While ( NOT All Layers Reach End Of Frame Sequence)
2   For each layer i=0th to Top Layer
3     While ( current NAL Unit index of layer ith  <=
4              current upper bound of layer ith
5                       AND
6             current NAL Unit index of layer ith <=  LastFrame
7           )
8           if (function 1 is true) send current NAL Unit of Layer ith )
9           THEN Increment current NAL unit index of layer ith
10    EndWhile
11    set LookAhead of layer ith using function 2
12    set current upper bound of layer ith to current upper bound of ith
          + LookAhead of layer ith
13
14  EndFor
15 EndWhile
```

Figure 5.12. Main section of adaptive look ahead scheduling algorithm

Function 1:

```
1 if  ( NAL Unit of layer (i-1)th is discarded by sender )   OR
2     ( Current time + estimate end to end delay time > estimate play-out
        deadline)
3 THEN Return false
4 Else Return true
5 End if
```

Figure 5.14 Function 1 of adaptive scheduling algorithm