Scalable Video Streaming over the Internet

A Thesis
Presented to
The Academic Faculty

by

Taehyun Kim

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

School of Electrical and Computer Engineering
Georgia Institute of Technology
January 2005

Scalable Video Streaming over the Internet

Approved by:

Date Approved: 10 January 2005

To my family

ACKNOWLEDGEMENTS

First of all, I would like to express my sincere gratitude to my advisor, Dr. Mostafa H. Ammar. During my doctoral studies at Georgia Institute of Technology, he has provided me with insightful vision, thoughtful guidance, and valuable comments. It has been a great pleasure to have him as an advisor. Without his guidance, this dissertation would not be successfully completed. Also I would like to express my gratitude to my co-advisor, Dr. Yucel Altunbasak for his invaluable comments and suggestions. I am also grateful to other members of my dissertation committee, Dr. John Barry, Dr. Chuanyi Ji, Dr. George Riley, Dr. Henry Owen, and Dr. Jack Brassil for their feedback and suggestions.

Special thanks go to Dr. Jack Brassil in HP Labs; Dr. Markus Hofmann and Dr. Volker Hilt in Lucent Bell Labs. They provided me with the opportunities to explore the multimedia research in the industrial environments. I appreciate their thoughtful consideration and constant encouragement that enabled me to have successful industry experiences.

I am also grateful to all other colleagues in the Networking and Telecommunications Group for their comments and our collaborations. I will cherish the pleasant memory of the excellent environment and friendly atmosphere in the group.

Finally, I would like to express the deepest gratitude to my family. Especially I would like to thank my wife and son, Ilhee and Hyunseong, for their love and encouragement so that I can devote myself to the doctoral research. I also appreciate our parents and brothers in Korea for their support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The objectives of this thesis are to investigate the challenges on video streaming, to explore and compare different video streaming mechanisms, and to develop video streaming algorithms that maximize visual quality. To achieve these objectives, we first investigate scalable video multicasting schemes by comparing layered video multicasting with replicated stream video multicasting. Even though it has been generally accepted that layered video multicasting is superior to replicated stream multicasting, this assumption is not based on a systematic and quantitative comparison. We argue that there are indeed scenarios where replicated stream multicasting is the preferred approach.

We also consider the problem of providing perceptually good quality of layered VBR video. This problem is challenging, because the dynamic behavior of the Internet's available bandwidth makes it difficult to provide good quality. Also a video encoded to provide a consistent quality exhibits significant data rate variability. We are, therefore, faced with the problem of accommodating the mismatch between the available bandwidth variability and the data rate variability of the encoded video. We propose an optimal quality adaptation algorithm that minimizes quality variation while at the same time increasing the utilization of the available bandwidth.

Finally, we investigate the transmission control protocol (TCP) for a transport layer protocol in streaming packetized media data. Our approach is to model a video streaming system and derive relationships under which the system employing the TCP protocol achieves desired performance. Both simulation results and the Internet experimental results validate this model and demonstrate the buffering delay requirements achieve desired video quality with high accuracy. Based on the relationships, we also develop realtime estimation algorithms of playout buffer requirements.

CHAPTER 1

INTRODUCTION

The growth of popular web sites serving multimedia contents has led to the increase of video streaming applications. However, video streaming over the Internet is complicated by a number of factors: 1) the Internet provides only the best effort service, and nothing is guaranteed about bandwidth, delay, and packet loss rate; 2) it is difficult to predict the bandwidth, delay, and loss rate information, since it is unknown and time-varying; 3) the heterogeneity of receiver capabilities is a significant problem when video streams are distributed over a multicast network; and 4) a congestion/flow control mechanism has to be employed to avoid the congestion collapse of the Internet.

This research is devoted to investigate the challenges on video streaming, to explore and compare different video streaming mechanisms, and to develop video streaming algorithms that maximize visual quality. To achieve these objectives, we consider scalable video streaming techniques that supports a number of receivers in a dynamic environment.

We first investigate scalable video multicasting. The heterogeneity of the Internet's transmission resources and end system capability makes it difficult to agree on acceptable traffic characteristics among the multiple receivers of a multicast video stream. Three basic approaches have been proposed to deal with this problem: 1) multicasting the replicated video streams at different rates, 2) multicasting the video encoded in cumulative layers, and 3) multicasting the video encoded in non-cumulative layers. Even though there is a common belief that the layering approach is better than the replicated stream approach, there have been no studies that compare these schemes. This research is devoted to such a systematic comparison. Our starting point is an observation (substantiated by results in the literature) that a bandwidth penalty is incurred by encoding a video stream in layers. We argue that a fair comparison of these schemes needs to take into account this penalty as well as the specifics of the encoding used in each scheme, protocol complexity, and the topological placement of the video source and the receivers relative to each other. Our results show that the believed superiority of layered multicast transmission relative to replicated stream multicasting is not as clear cut as is widely believed and that there are indeed scenarios where replicated stream multicasting is the preferred approach.

Second, we consider the problem of layered variable bit rate (VBR) video streaming. The dynamic behavior of the Internet's transmission resources makes it difficult to provide perceptually good quality streaming video. Scalable video encoding techniques have been proposed to deal with this problem. However, an encoded video generally exhibits significant data rate variability to provide consistent visual quality. We are, therefore, faced with the problem of accommodating the mismatch between the available bandwidth variability and the encoded video variability. In this research, we investigate quality adaptation algorithms for scalable encoded VBR video over the Internet. Our goal is to develop a quality adaptation scheme that maximizes perceptual video quality by minimizing quality variation while at the same time increasing the usage of available bandwidth. We propose an optimal adaptation algorithm and a realtime adaptation algorithm based on whether the network conditions are known *a priori*. Experimental results show that the realtime adaptation as well as the optimal adaptation algorithm provide consistent video quality when used over both TFRC and TCP.

Finally, we investigate transport layer protocol for packetized media delivery for video streaming. TCP is one of the most popular transport protocols for video streaming, even though the rate variability of TCP makes it difficult to provide good video quality. To accommodate the variability, video streaming applications require receiver-side buffering. In current practice, however, there are no guidelines for the provisioning of the receiver buffer, and smooth playout is insured through over-provisioning. In this work, we are interested in memory-constrained devices where it is important to determine the right size for the playout buffer in order to insure a prescribed video quality. To that end, we characterize video streaming over TCP in a systematic and quantitative manner. We first model video streaming system analytically and derive expressions of buffer size requirements based on the model. Our model takes account of three scenarios: 1) when TCP throughput matches video encoding rate, 2) when TCP throughput is smaller than the encoding rate, and 3) when TCP throughput is limited by the maximum window size. Experimental results of both simulations and the Internet experiments validate the model and demonstrate that the minimum buffering delay achieves desired video quality.

We also develop realtime estimation algorithms of playout buffer requirements in video streaming over TCP. Based on the desired buffer size relationship, we first develop a realtime estimation algorithm of receiver buffer size by estimating the packet loss rate and round-trip time. We further develop realtime throughput adaptation algorithms that dynamically adjust data transmission rate to prevent playback disruptions proactively. Experimental results show that proposed algorithms coupled with an error resilient video codec reduce the quality variability and improve overall video quality significantly.

This dissertation is organized as follows. Chapter 2 addresses the problem of scalable video multicasting and compares the replicated stream video multicasting, the cumulatively layered video multicasting, and the non-cumulatively layered video multicasting. In Chapter 3, we develop quality adaptation algorithms for scalable encoded VBR video streaming over the Internet. Chapter 4 considers the video streaming model over TCP and develops the buffer size requirements at a receiver. Based on the relationships, we develop realtime estimation algorithms of playout buffer requirements in Chapter 5. This dissertation is concluded in Chapter 6.

CHAPTER 2

SCALABLE VIDEO MULTICASTING

2.1 Motivation

A system for multicasting video over the Internet has to deal with the question of heterogeneity of the receivers capability and/or requirements. Typically, receivers and the paths leading to them will have different reception capacity. We are, therefore, faced with the problem of trying to accommodate this difference among the receivers: low capacity receivers are heavily loaded and suffer from network congestion, but high capacity receivers are lightly loaded and under-utilized. This problem has been addressed by many researchers (e.g., [8], [9], [11], [34], [49], [51]). Note that the problem of multicasting video in a heterogeneous data rate environment is important regardless of whether native network layer multicast or application layer multicast [12] are used.

There are three basic approaches:

- **The replicated stream approach** [11], [34]

  In this approach, the video source multicasts several streams with identical content but at different data rates. Each stream is multicast over its own multicast group. Receivers subscribe to the appropriate stream and may switch among streams as their capacity changes. These streams can be obtained by encoding the source video with different compression parameters.

- **The cumulative layering approach** [49], [51]

  In this approach, the video is encoded in a base layer and one or more enhancement layers. The base layer can be decoded independently, but the enhancement layers can be decoded cumulatively (i.e., layer k can only be decoded along with layers 1 to $k-1$). The enhancement layers contribute to the improvement of the video quality that leads to the progressive refinement. Each layer is multicast on its own group by a source. Receivers join at least the layer 1 multicast group and add/drop other layers according to their reception capacity.

MPEG-2, MPEG-4, and H.263 support cumulatively layered encoding by defining the scalability modes, such as data partitioning, SNR scalability, spatial scalability, and temporal scalability [30], [31], [32]. A combination of the scalability modes can lead to hybrid scalability consisting of multiple layers.

- **The non-cumulative layering approach** [9], [26]

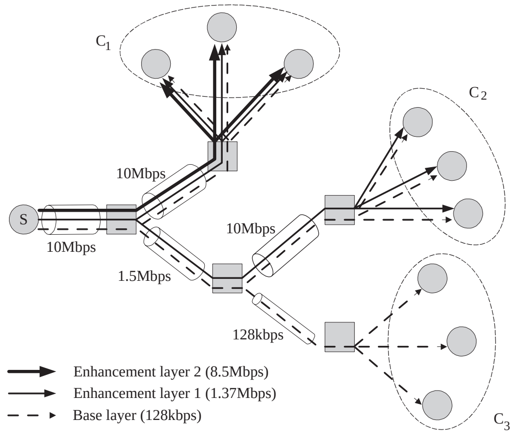
  In this approach, the video is encoded in two or more independent layers. Each layer can be decoded independently and provide improvements to the video reception quality. Each receiver can join any subset of the video layers.

  Multiple description coding (MDC) can be used for non-cumulatively layered video multicasting. Each description in MDC can lead to the reconstruction of the source video, and multiple descriptions together yield a construction with the smallest distortion [17]. To provide these features, a number of MDC schemes for video encoding have been developed recently, based on predictive MD quantizer [72], MD transform coding [60], and multiple states [1]. Each scheme provides different characteristics of compression efficiency, delay, and error resilience.

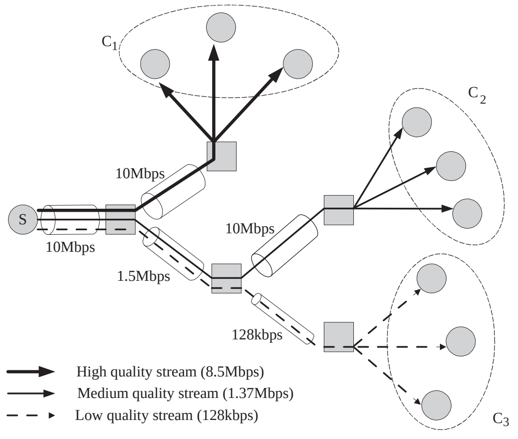
There is a common belief that the layering approach is better than the replicated stream approach. The main argument is that replicated streams waste bandwidth by duplicating the transmission of the content represented by the base layer (and possibly other lower layers). Even though this is a widely stated conclusion, we are not aware of any studies that actually compared these approaches in a quantitative and systematic manner [38], [39].

The goal of this research is to compare these video multicast techniques. Our starting point is an observation (substantiated by results in the literature) that by encoding a video stream in layers, one incurs a bandwidth overhead [19], [21], [35], [44], [62]. This overhead can sometimes change the bandwidth efficiency in favor of replicated stream video multicasting. We argue that a fair comparison of these schemes needs to take into account this overhead as well as the specifics of the encoding used in each scheme, protocol complexity, and the topological placement of the video source and the receivers relative to each other.

Consider an example of the scalable video multicasting shown in Figure 1. A video server is run on S and three sets of receivers, C_1, C_2, and C_3 are connected to the server from different domains.



(a) Layered video multicasting


C1
C2
C3
S
10Mbps
10Mbps
10Mbps
1.5Mbps
128kbps
High quality stream (8.5Mbps)
Medium quality stream (1.37Mbps)
Low quality stream (128kbps)

(b) Replicated stream video multicasting

Figure 1: Illustration of the scalable video multicasting. The received data rate of layered video multicasting is greater than that of replicated stream video multicasting. However, if we take account of layering overhead, layered video multicasting does not always provide better perceived visual quality.

Figure 1 (a) shows how layered video multicasting accommodates the heterogeneity problem. Since the receivers in C_1 have high link capacity, they can subscribe to all layers. However, if any of the receivers in C_2 or C_3 tries to join the second enhancement layer, the 1.5Mbps link becomes congested which leads the receiver to leave the second enhancement layer. Also, the receivers in C_3 have to join only the base layer because of the 128kbps link. Hence, the reception rates of C_1, C_2, and C_3 are 10Mbps, 1.5Mbps, and 128kbps.

On the other hand, Figure 1 (b) illustrates the replicated stream video multicast scheme. Since the receivers in replicated stream multicasting subscribe to only one stream, the receivers in C_1 have to join one of the three streams and consequently they subscribe to the high quality stream over the 10Mbps link. However, a receiver in C_2 or C_3 cannot receive the high quality stream because of the 1.5Mbps bottleneck link: if the receiver tries to subscribe to the high quality stream, the 1.5Mbps link will be congested. In the same way, a receiver in C_3 receives the low quality stream due to the limited capacity of the 128kbps link. Hence, the reception rates of C_1, C_2, and C_3 are 8.5Mbps, 1.37Mbps, and 128kbps.

Unfortunately, the data reception rate does not account for the video reception quality. Assume that layered video multicasting incurs 20% of layering overhead, the data rates contributing to the video quality in the layering scheme are 8Mbps, 1.2Mbps, and 102.4kbps for C_1, C_2, and C_3. Compared with the reception rate of the replicated stream scheme, the average data rate of the layering scheme contributing to the video quality is smaller. Hence, we can expect in this case that the average video quality of layered video multicasting is not better than that of replicated stream video multicasting.

The amount of overhead in these schemes will of course depend on the specifics of the encoding of the layering and replicated stream data (i.e., the number and rates of the streams). A fair comparison between these schemes needs to also take into account the protocol complexity as well as the topological placement of the video source and the receivers relative to each other.

This chapter is devoted to such a comparison. It is organized as follows. In Section 2.2, we consider the issue of layering overhead in more detail. Section 2.3 considers the question of optimizing the rate allocation to layers and to replicated streams. This optimization is necessary to insure that a fair comparison of the best layering scheme with the best replication scheme. Section 2.4 and 2.5 report on experiment results we have used to provide a quantitative comparison. Section 2.6 provides a comparison of the protocol overheads involved in the multicast schemes. Section 2.7 concludes this chapter.

2.2 Overhead in Layered Video

In this section, we describe how layered encoding of video incurs a bandwidth overhead. Consider a video that is encoded as a single (non-layered) stream with a given quality and that results in a data rate of R_{nl}, including all protocol/packetization overheads. Let the same video be encoded in m cumulative layers with the data rate for layer i being R_{l_i}, again including all protocol/packetization overhead. We further assume that the layered encoding of the video is such that, if a receiver receives and decodes *all* layers, the quality of the video will be the *same* as the non-layered video stream with rate R_{nl}.

The basic conclusion that we reach is that

$$R_{nl} \leq R_l = \sum_{i=1}^{m} R_{l_i}.$$

Results in the literature indicate that the equality above is rarely achieved and that R_l can be as much as 20%–30% higher than R_{nl}.

We substantiate this conclusion as follows:

- **Information theoretic results**

  These results are derived in terms of the *rate distortion function*, $R(P, \Delta)$, which describes the required rate to encode a memoryless source at a maximum distortion of Δ, where the distortion is a measure of the quality degradation represented by the encoding of the source.

  The general result (described more formally in Appendix A) is that, for the same source and the same distortion, a *successively refined* (i.e., layered) encoding requires at least as much data rate as a non-layered encoding. While equality is possible, it requires a strict Markovian condition to apply to the source and is generally not achievable. Moreover, the result in [35] shows that the performance of the layered encoding is not better than that of non-layered encoding for a finite length block code, even if the Markovian condition holds.

- **Packetization overhead**

  For certain scalability modes in the standards, enhancement layers are designed to be syntactically independent of one another. Along with the residual information for every enhancement layer, the data stream needs to also carry syntactic data, such as picture header, start codes, GOP information, and macroblock header. This can incur a large amount of overhead especially at low data rates [44].

- **Experimental evidence**

  Figure 2 shows an experimental result of the video quality versus data rates for the *flower* sequence by comparing MPEG-2 SNR scalability and non-scalability mode.

The video quality is measured in PSNR by varying quantization step size. A layered stream has two layers consisting of a base layer and an enhancement layer at (Q_b, Q_e), where Q_b is the quantization step size of the base layer and Q_e is that of the enhancement layer. Both PSNR and data rate are averaged over the entire video.

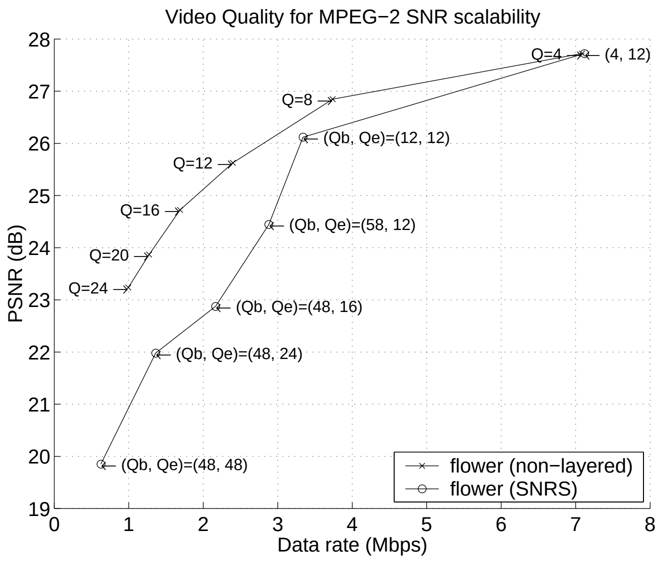


Figure 2: Performance comparison of MPEG-2 SNR scalability and non-scalability mode using a software MPEG-2 codec. The layering overhead ranges from 0.4% at 27.7dB to 117% at 23.2dB.

This result demonstrates that a layered stream requires more data rate than a non-layered stream to provide the same quality. The difference in data rate ranges from 0.4% at 27.7dB to 117% at 23.2dB. Note that the difference is expected to grow as the number of layers increases, since the accumulation of the redundancy leads to the increase of the overall distortion in layered video encoding [73].

Similar and more extensive experimental results can be found in [43]. The authors investigated the impact of the number of layers, bit rates, and packet loss on the perceptual video quality as determined by subjects scoring the quality of the video, when MPEG-2 data partitioning and SNR scalability are employed. The experimental result on data partitioning shows that the difference ranges from nearly 0 for the highest quality video (scoring close to 4.5) to almost 57% for fair quality video (scoring close to 3). For a score of 4 (good quality video), the overhead varies from 2% (the *flower* sequence) to 49% (the *basket* sequence).

Recent research efforts have focused on developing an efficient scalable video encoding. One of the most significant results is MPEG-4 fine grained scalability (FGS) [59]. Several features have been employed to achieve efficient scalability, such that 1) it accommodates a wide range of data rate variability by distributing enhancement layers over a wide range of bit rates, 2) it provides an efficient coding based on bit-plane coding that is more efficient than run level coding, and 3) it separates the FGS layer from the motion compensation stage to eliminate drift in the enhancement layer.

Extensive experimental results can be found in [59]. The authors compared the performance of FGS with the traditional SNR scalability, and it was shown that FGS provides better performance than SNR scalability. They also compared the performance of FGS with non-scalability mode. However, it was found that FGS suffers coding efficiency if a video sequence has a high temporal correlation, or if the data rate of base layer is small. Experimental results show that there exists a coding overhead in FGS, and the overhead ranges from nearly 0 at a sequence of high degree of motion to 50% at low degree of motion.

- **Protocol overhead**

  The nature of the subscription to multiple layers in layered video multicasting may cause some additional overhead. Since a receiver generally subscribes to more than one layers in layered video multicasting, the receiver needs to manage the subscription to multiple layers. For example, in the context of receiver-driven layered multicast (RLM) [51], the probing mechanism of available bandwidth depends on join experiments. However, join experiments incur bandwidth overhead since they require to send a join message and to multicast a message for shared learning. The amount of bandwidth overhead is increased, as the group size of a multicast group grows. Also the subscription of multiple layers requires more buffer size and better synchronization capability than replicated stream video multicasting. More discussions and experimental results are presented in Section 2.6.

- **Error control overhead**

  When video layers/streams are delivered over the Internet, some packets may be lost due to network congestion. Since the packet loss causes a degradation of visual quality, error control mechanisms for layered video multicasting can be employed:

  1) *Forward error control (FEC) based* mechanisms have been developed to reconstruct the original information by adding extra redundancy. The authors in [70] proposed a layered FEC scheme by providing error control layers for cumulatively layered video multicasting to achieve the desired level of protection. However, this scheme incurs some amount of overhead to process the redundancy. Note that FEC schemes can be employed for replicated video streaming [7].

  2) *Path diversity* can allow different layers to be delivered over different routes. Combined with path diversity, non-cumulatively layered video multicasting exhibits better video quality as long as packet loss occurs only over disjoint channels. The authors in [2] investigate the effect of path diversity and compare the performance of non-cumulative layering with that of replicated video streaming. Experiment results show that the number of disjoint links should be large enough to achieve better performance than replicated video streaming. However, this mechanism incurs an operating overhead: path diversity can be of benefit only if the multicast trees of different video layers are disjoint. It therefore requires that multiple servers should be placed in different domains with each of the servers multicasting a different video layer.

2.3 Optimizing Stream Rates

To carry out a fair comparison of the layered and replicated stream multicast schemes, we need to insure that each scheme is optimal. We also need to insure that a similar set of rate choice is available for all schemes. The question here is how to determine the number and rates for the set of replicated streams and for the layers.

In this section, we shall present 1) a *rate allocation* algorithm to determine the data rate of each stream and 2) a *stream assignment* algorithm to determine the reception rate of each receiver by aggregating the data rates of the assigned streams in layered and replicated stream multicasting. The goal of these algorithms is to maximize the bandwidth utilization by each scheme for a given network, a particular set of receivers, and given available bandwidth on the network links.

To this end, we model the network by a graph $G = (V, E)$, where V is a set of vertices representing routers and hosts. E is a set of edges representing connection links which is defined over $V \times V$. A set of receivers is defined by $C = \{c_i \mid c_i \in V,\ i = 1, \ldots, n\}$, where n is the number of receivers.

An *isolated rate* for each receiver is defined by the data reception rate of the receiver if there is no constraint from other receivers in the same session [34]. The isolated rate can be computed by the Dijkstra's algorithm.

A bandwidth function $B : E \rightarrow \mathbf{R}^+$ is defined on E with $b_j = B(e_j)$, where $\mathbf{R}^+$ is the set of positive real numbers. The bandwidth function is considered as a measure of the residual bandwidth available on the link e_j.

2.3.1 Cumulatively Layered Multicasting

2.3.1.1 Rate Allocation

A cumulatively layered multicast session is defined by $\alpha = \{\alpha_i \mid \alpha_i \in \mathbf{R}^+,\ i = 1, \ldots, m\}$, where α_i is the data rate of the ith layer stream and m is the number of layers.

The first rate allocation algorithm for cumulatively layered video multicasting was proposed in [68] by maximizing the average signal reception quality. The authors in [76] proposed an optimal receiver partitioning algorithm to determine the optimal stream rates using dynamic programming. We adopt the latter algorithm to determine the optimal data rates of α_i: we define the group utility function by $U(\{j+1, \ldots, i\}) = (i-j)\, f(r_{j+1})$, where r_{j+1} is the isolated rate of the receiver $j + 1$ and $f(r_{j+1})$ is an effective rate allocation function. The *effective rate allocation function* is defined by $f : \mathbf{R}^+ \rightarrow \mathbf{R}^+$ describing the *effective reception rate* which is the data reception rate contributing to video quality. By applying the optimal receiver partitioning algorithm with this group utility function, we can maximize the overall effective reception rate.

2.3.1.2 Stream Assignment

The stream assignment algorithm is presented in Figure 3, when the stream rates α_i are given. We assume that each layer is routed over the same path and that each receiver can join as many layers as possible within the isolated rate (line 2). Hence, the reception rate is determined by the sum of stream rates that does not exceed the isolated rate.

```
1:  Compute the isolated rates
2:  Assign $\Sigma_i \alpha_i$ that does not exceed the isolated rate
```

Figure 3: Stream assignment algorithm for cumulatively layered multicasting. A receiver can subscribe to as many layers as possible within its capability.

2.3.2 Replicated Stream Multicasting

2.3.2.1 Rate Allocation

A replicated stream multicast session is defined by $\beta = \{\beta_i \mid \beta_i \in \mathbf{R}^+,\ i = 1, \ldots, m\}$, where β_i is the data rate of a replicated stream and m is the number of replicated streams.

We determine the stream rates based on α_i, which is determined in Section 2.3.1.1. Specifically, β_1 corresponds to the base layer of cumulative layering and the other stream rates are determined in a cumulative manner: if a receiver can join up to k layers in cumulative layering, the receiver has the capability to join a replicated stream of data rate $\beta_k = \sum_{i=1}^{k} \alpha_i$.

2.3.2.2 Stream Assignment

In the stream assignment algorithm, it is required that the reception rate of every receiver is strictly greater than zero so that there is no receiver that cannot receive any stream. This requirement can be satisfied by multicasting β_1 to all receivers, and therefore a receiver can subscribe to either the base layer stream or a higher rate stream.

We define δ_i, such that

$$\delta_i = \begin{cases} \beta_1, & i = 1 \\ \beta_i - \beta_1, & 1 < i \leq m. \end{cases}$$

The stream assignment algorithm for replicated stream multicasting determines the data reception rate of a receiver, given the set of data rates $\delta = \{\delta_i \mid \delta_i \in \mathbf{R}^+,\ i = 1, \ldots, m\}$.

We define Ω_i to be a set of receivers, such that $\Omega_i = \{c_j \mid \phi_\delta(c_j) = \delta_i\}$, where ϕ_δ is the rate allocation function defined by $\phi_\delta : C \rightarrow \delta$.

We set up two objectives for stream assignment.

1. The minimum reception rate of all receivers is strictly greater than zero.

2. Maximize $Z_\delta = \sum_{i=1}^{m} |\Omega_i|\delta_i$ subject to $\sum_{i \in \Gamma_{e_j}} \delta_i \leq b_j$, where $\Gamma_{e_j} = \{i \mid e_j \in E_i\}$, $T_i = (V_i, E_i)$, and T_i is a multicast tree for a replicated stream i.

We develop a greedy algorithm to achieve these requirements. The algorithm is described in Figure 4. We first allocate δ_1 to all receivers to satisfy the minimum reception rate constraint (lines 2–7). Next, a receiver is assigned a stream that has not yet been assigned and has a maximum value of the product of the group size and the effective reception rate until *every receiver* is assigned to at least one stream (lines 8–18). In stream selection, we assign an identity function to the effective rate allocation function for replicated stream multicasting, since a non-layered stream is not supposed to incur layering overhead. Therefore, every receiver subscribes to either high quality stream (line 12) or the base layer stream (line 15).

```
1:  Compute the isolated rates
2:  if all isolated rates are greater than $\delta_1$ then
3:      $b_j \leftarrow b_j - \delta_1$, where $b_j = B(e_j)$ and $e_j \in E$
4:      $\delta \leftarrow (\delta \backslash \delta_1) \cup \{0\}$
5:  else
6:      There is no feasible solution
7:  endif
8:  while not $\Omega = \emptyset$ do
9:      Compute the isolated rates
10:     Select a stream $i$ with $|\Omega_i|\delta_i = \max_j |\Omega_j|\delta_j$ $(\delta_i, \delta_j \in \delta)$
11:     if $\delta_i > 0$ then
12:         Assign $\delta_i$ to $\Omega_i$
13:         Reduce the link capacity leading to $\Omega_i$
14:     else
15:         Assign $\delta_1$ to $\Omega_i$
16:     endif
17:     $\Omega \leftarrow \Omega \backslash \Omega_i$
18: enddo
```

Figure 4: Stream assignment algorithm for replicated stream multicasting. A receiver can subscribe to either the base layer stream or high quality stream.

The feasibility of the stream assignment algorithm is guaranteed, if the data rate of a replicated stream multicasting session is determined by the rate allocation algorithm in Section 2.3.2.1. This is because the data rate of the base layer is originally determined by the optimal partitioning algorithm. Otherwise, this algorithm may not provide a feasible solution when any receiver has an isolated rate smaller than δ_1.

Note that the rate allocation scheme and the stream assignment algorithm may reduce the data reception rate of a receiver compared to cumulatively layered multicasting. However, this does not always guarantee the effective reception rate of cumulatively layered multicasting is greater than that of replicated stream multicasting, when we take the layering overhead into account.

2.3.3 Non-cumulatively Layered Multicasting

2.3.3.1 Rate Allocation

A non-cumulatively layered multicast session is defined by $\gamma = \{\gamma_i \mid \gamma_i \in \mathbf{R}^+,\ i = 1, \ldots, m\}$, where γ_i is the data rate of a non-cumulatively layered stream and m is the number of streams. A set of receivers assigned to the stream i is defined by $\Omega'_i = \{c_j \mid \gamma_i \in \phi_\gamma(c_j)\}$, where ϕ_γ is the stream rate function defined by $\phi_\gamma : C \to P(\gamma)$ and $P(\gamma)$ is a power set of γ. The set of all receivers is $\Omega' = \cup_{i=1}^{m} \Omega'_i$.

In non-cumulatively layered multicasting, a receiver can subscribe to any subset of layers. This property provides a fine granularity for non-cumulatively layered multicasting. For example, given a non-cumulatively layered stream $\gamma = \{1, 2, 4\}$, a heterogeneity resulting from seven different isolated rates of $\{1, 2, 3, 4, 5, 6, 7\}$ can be accommodated through selective subscription. Hence, the non-cumulatively layered session γ shows the same performance as the cumulatively layered session $\alpha = \{\alpha_i | \alpha_i = 1,\ i = 1, \ldots, 7\}$ (e.g., $\alpha_1 + \alpha_1 + \alpha_1$ can be achieved by $\gamma_1 + \gamma_2$). This example demonstrates that the heterogeneity caused by $\sum_{i=1}^{m} \binom{m}{i} = 2^m - 1$ different link capacities can be accommodated by aggregating the reception rates of m non-cumulative layers. The work in [9] describes a fine grained layered multicasting based on this property. The authors propose a fine grained rate adjustment scheme through at most three join/leave operations.

In this section, we propose a rate allocation algorithm for non-cumulatively layered streams. The stream rates γ_i are allocated based on α_i as follows:

$$\begin{aligned}
\gamma_1 &= \alpha_1, \\
\gamma_2 &= \alpha_1 + \alpha_2, \\
\gamma_1 + \gamma_2 &= \alpha_1 + \alpha_2 + \alpha_3, \\
\gamma_3 &= \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4, \\
&\cdots \\
\gamma_2 + \gamma_3 + \cdots + \gamma_m &= \alpha_1 + \alpha_2 + \cdots + \alpha_{2^m-2}, \\
\gamma_1 + \gamma_2 + \gamma_3 + \cdots + \gamma_m &= \alpha_1 + \alpha_2 + \cdots + \alpha_{2^m-2} + \alpha_{2^m-1},
\end{aligned}$$

where the optimum α_i can be determined in Section 2.3.1.1.

We simplify this relationship in a matrix form: $\mathbf{AX} = \mathbf{BY}$, where $\mathbf{A}$ is a binary counting matrix, $\mathbf{B}$ is a lower triangular matrix, $\mathbf{X}$ is a vector of the allocated data rates of non-cumulative layering, and $\mathbf{Y}$ is a vector of the optimal data rates of cumulative layering, such that

$$\mathbf{A} = \begin{pmatrix} 0 & \cdots & 0 & 0 & 1 \\ 0 & \cdots & 0 & 1 & 0 \\ 0 & \cdots & 0 & 1 & 1 \\ 0 & \cdots & 1 & 0 & 0 \\ & \vdots & & & \\ 1 & \cdots & 1 & 1 & 1 \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} 0 & 0 & \cdots & 0 & 0 & 0 & 1 \\ 0 & 0 & \cdots & 0 & 0 & 1 & 1 \\ 0 & 0 & \cdots & 0 & 1 & 1 & 1 \\ 0 & 0 & \cdots & 1 & 1 & 1 & 1 \\ & \vdots & & & & & \\ 1 & 1 & \cdots & 1 & 1 & 1 & 1 \end{pmatrix},$$

$$\mathbf{X} = \begin{pmatrix} \gamma_m \\ \gamma_{m-1} \\ \gamma_{m-2} \\ \vdots \\ \gamma_2 \\ \gamma_1 \end{pmatrix}, \qquad \mathbf{Y} = \begin{pmatrix} \alpha_{2^m-1} \\ \alpha_{2^m-2} \\ \alpha_{2^m-3} \\ \vdots \\ \alpha_2 \\ \alpha_1 \end{pmatrix}.$$

However, it is not generally possible to determine the data rate γ_i for given α_i, since the number of equations exceeds that of unknown variable γ_i. We develop an approximate rate allocation scheme by minimizing the mean square error, $Z = (\mathbf{AX} - \mathbf{BY})^T(\mathbf{AX} - \mathbf{BY})$. The allocated data rates of non-cumulatively layered multicast streams are determined by $\mathbf{X} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{BY}$, since

$$\begin{aligned} \nabla_X Z &= \nabla_X(\mathbf{AX} - \mathbf{BY})^T(\mathbf{AX} - \mathbf{BY}) \\ &= 2\mathbf{A}^T\mathbf{AX} - 2\mathbf{A}^T\mathbf{BY} \\ &= 0, \end{aligned}$$

where ∇_X is a gradient operator of $\mathbf{X}$.

2.3.3.2 Stream Assignment

We have two objectives to assign the non-cumulatively layered streams.

1. The minimum reception rate of all receivers is strictly greater than zero.
2. Maximize $Z_\gamma = \sum_{i=1}^{m} |\Omega_i'| f(\gamma_i)$ when $\sum_{i \in \Gamma_{e_j}} \gamma_i \leq b_j$, where $\Gamma_{e_j} = \{i \mid e_j \in E_i\}$, $T_i = (V_i, E_i)$, T_i is a multicast tree for a non-cumulatively layered stream i, and $f(\gamma_i)$ is the effective rate allocation function.

In Figure 5, we present a greedy algorithm for non-cumulatively layered multicasting to assign a stream with a maximum value of the product of the group size and the effective reception rate. We first allocate the base layer stream, γ_1, to all receivers (lines 2–8). A receiver is assigned every layer which has not yet been multicast to any receiver and maximizes the product of group size and effective reception rates until *every stream* is assigned (line 11). The data reception rate of a receiver is the sum of data rates of all assigned streams, since the aggregated data rate of non-cumulatively layered streams leads to the minimum distortion.

It should be noted that we need an effective rate allocation function to determine the objective function in the algorithm, since it is desirable to maximize the data reception rate contributing to the video quality instead of the data rate itself. In Section 2.4.2, we propose three types of effective rate allocation functions by modeling layering overhead.

```
1:  Compute the isolated rates
2:  if all isolated rates are greater than $\gamma_1$ then
3:      Assign $\gamma_1$ to all receivers
4:      $b_j \leftarrow b_j - \gamma_1$, where $b_j = B(e_j)$ and $e_j \in E$
5:      $\gamma \leftarrow (\gamma \backslash \gamma_1) \cup \{0\}$
6:  else
7:      There is no feasible solution
8:  endif
9:  while not $\gamma = \emptyset$ do
10:     Compute the isolated rates
11:     Select a stream $i$ with $|\Omega'_i| f(\gamma_i) = \max_j |\Omega'_j| f(\gamma_j)$ $(\gamma_i, \gamma_j \in \gamma)$
12:     if $\gamma_i > 0$ then
13:         Add $\gamma_i$ to $\Omega'_i$
14:         Reduce the link capacity leading to $\Omega'_i$
15:     endif
16:     $\gamma \leftarrow \gamma \backslash \gamma_i$
17: enddo
```

Figure 5: Stream assignment algorithm for non-cumulatively layered multicasting. A receiver can subscribe to multiple streams. The data rate of the aggregated streams leads to the minimum distortion.

2.4 Models in Experiments

We compare the performance of the video multicast schemes by experiment. The main goal in the experiment is to evaluate the impact of the parameters, including the amount of layering overhead, the number of layers, and the topological placement of receivers, on the video reception quality. All schemes use the rates and stream assignment as determined in Section 2.3.

2.4.1 Network Model

We use GT-ITM [77] to generate 100 different transit-stub graphs representing hierarchical Internet topologies. The graphs consist of 1,640 nodes including 10 transit domains, 4 nodes per transit domain, 4 stubs per transit node, and 10 nodes in a stub domain (i.e., the number of nodes is $1640 = 10 \cdot 4 \cdot (1 + 4 \cdot 10)$). We assign 2.4Gbps to transit-to-transit edges; 10Mbps and 1.5Mbps to stub-to-stub edges; and 155Mbps, 45Mbps, and 1.5Mbps to transit-to-stub edges. The available link bandwidth is chosen uniformly in the range 1%-80% of the full capacity of the edge.

2.4.2 Layering Overhead Models

Unless otherwise mentioned, the number of cumulatively layered video streams and the number of replicated video streams are 8, and the number of non-cumulatively layered video streams is 4 as discussed in Section 2.3.3.1. The amount of overhead incurred by cumulative and non-cumulative layering is modeled in three ways:

- **Proportional model**

  In the proportional model, we assume that the amount of layering overhead is affected proportionally by the amount of the data reception rate for the sake of simplicity. Based on the observations in Section 2.2, layering overhead is chosen to be 25% which is an average value to deliver a good quality video in MPEG-2. Since layering overhead tends to increase as the number of layers grows [62], 25% layering overhead is a relatively optimistic value for MPEG-2 scalability: the experimental results of MPEG-2 data partitioning in [43] show that, if the video quality of the *flower* sequence is compared, 2-layer stream incurs 245kbps overhead, whereas 3-layer stream incurs 460kbps overhead.

- **Dynamic model**

  The proportional overhead model assumes the same amount of layering overhead for the same data reception rate. However, the amount of layering overhead depends on many parameters in practice, such as the specifics of encoder, the degree of motion, and/or the dynamic range of the bandwidth (i.e., $[R_1, R_l]$, where R_1 is the data rate of base layer and R_l is the data rate of all layers).

  The dynamic overhead model captures the notion of the dynamically varying nature of the layering overhead. The model is based on the observations of MPEG-4 FGS in [59]. The authors showed that the *stefan* sequence exhibiting high temporal correlation between frames (i.e., low degree of motion) incurs bandwidth overhead, since FGS exploits the temporal information at the base layer only.

  The amount of layering overhead increases when the data rate of base layer is small (i.e., wide dynamic range). The experiment results demonstrate that there exists

200kbps layering overhead when the data rate of base layer is 200kbps, and 40kbps overhead when base layer is 300kbps. Based on the observations, we model the layering overhead by the linear interpolation:

$$\text{Layering overhead } = 520 - 1.6 \cdot R_{l_1} \text{ (kbps)},$$

where R_{l_1} is the data rate of the base layer and $R_{l_1} \leq 325$kbps.

- **Low overhead model**

  The low overhead model assumes that there is no layering overhead in a layered stream and that the whole received data contributes to the improvement of video quality. The theoretical result in Section 2.2 shows that the data rate of a layered stream can be the same as that of a non-layered stream providing the same quality, when the Markovian condition holds. Although it is hard to achieve, the overhead of layered encoding can be close to zero.

  Examples of this scenario can be found in several wavelet-based video codecs that have been developed recently [37], [63], [64]. These codecs are currently considered in the research community but are not widely adopted video coding standards yet.

  Another example is the MPEG-4 FGS codec in Section 2.2 that exhibits very low layering overhead for sequences with high degree of motion. Authors in [59] demonstrate that the *flying* sequence exhibiting low temporal correlation shows a high coding efficiency and the performance of FGS is similar to that of the non-scalable case. Note that, even for a sequence of low degree of motion, the FGS codec also shows low layering overhead when the dynamic range of data rate is changed. Experimental results from the *stefan* sequence shows that the layering overhead is close to zero when the data rate of the base layer is greater than 500kbps.

  It should be noted that the low overhead model not only assumes that the coding overhead is close to zero but also assumes that other layering overheads, such as the packetization overhead, the protocol overhead, and the error control overhead are also close to zero.

2.4.3 Performance Measures

In the experiment, we compare the performance by using the following measures:

- The *average reception rate* which is the average data rate received by a receiver
- The *average effective reception rate* where the effective reception rate at a receiver is defined by the amount of data received less the layering overhead
- The *total bandwidth usage* calculated by adding the total traffic carried by all links in the network for the multicast session – including all layers and all replicated streams
- The *efficiency* defined by

$$\text{efficiency} = \frac{\text{total effective reception rate}}{\text{total bandwidth usage}}.$$

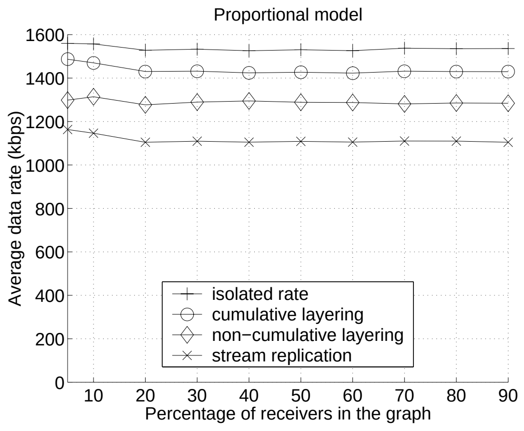
  The efficiency is a measure of delivered data rate contributing to the video quality for each unit of bandwidth used in the network.

2.5 Experiment Results

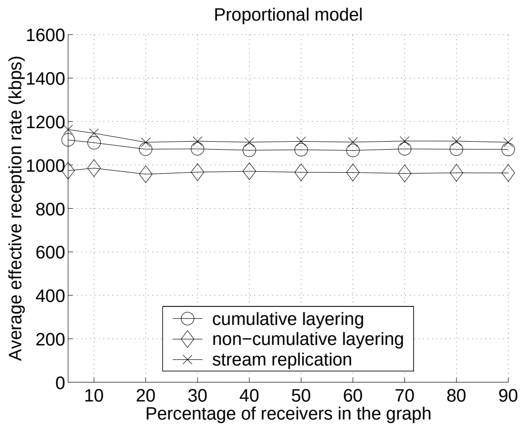
2.5.1 Random Distribution

In the first experiment, we randomly select a server and receivers from a set of nodes in the graph. Receivers are selected from all domains, which results in random distribution of receivers. We investigate the performance of the video multicast schemes by varying the number of receivers.

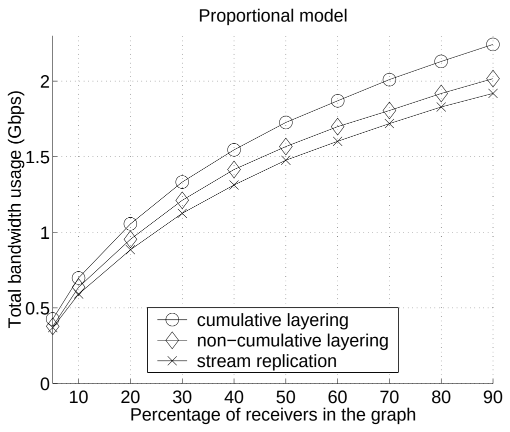
Figure 6 (a) shows the average data reception rate in the proportional model. The average reception rate of cumulative layering is the largest in this scenario: the average reception rate of cumulative layering, non-cumulative layering, and replicated stream video multicasting are as much as 92%, 84%, and 73% of the isolated rate. Figure 6 (b) shows the average effective reception rate of all receivers. The effective reception rate of replicated stream video multicasting is the largest in the experiment. Since the video quality is improved as the effective data rate increases, we expect that the average video quality in replicated stream video multicasting is the best in this model. The total bandwidth usage



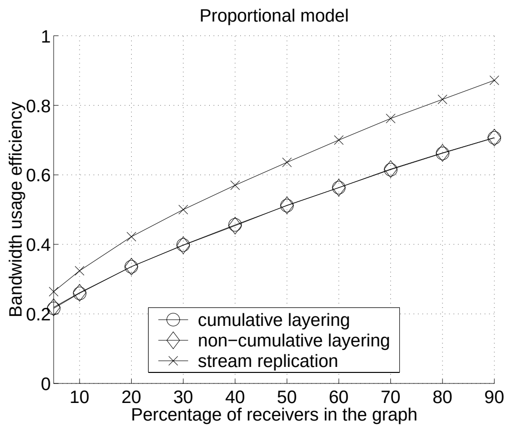
(a) Reception rate



(b) Effective reception rate



(c) Total bandwidth usage



(d) Efficiency

Figure 6: Experiment results in the proportional model under the random receiver distribution. (a) shows that the average received data rate of cumulatively layered multicasting is the largest. However, (b) shows that the average effective reception rate of replicated stream multicasting is the largest because of the layering overhead. Replicated stream multicasting also exhibits the best bandwidth usage efficiency.

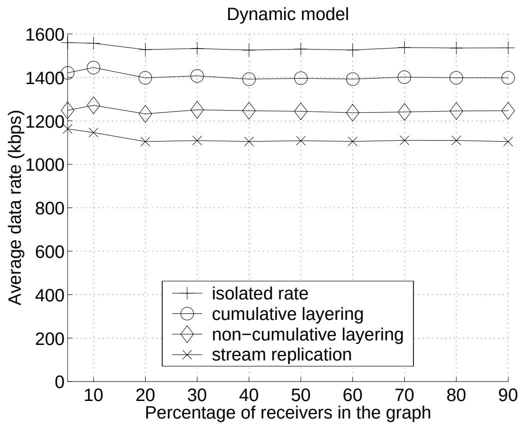
is presented in Figure 6 (c). The bandwidth consumption of cumulatively layered video multicasting is the largest and that of replicated stream video multicasting is the smallest. The efficiency of the video multicast schemes are demonstrated in Figure 6 (d). Although cumulative and non-cumulative layering have good bandwidth scalability and high reception rates, these schemes are required to pay layering overhead. This makes replicated stream video multicasting the most efficient.

Figure 7 shows the experiment results of the video multicast schemes in the dynamic overhead model. These results exhibit similar behavior to the proportional model in Figure 6. In Figure 7 (a), the average reception rate of cumulative layering, non-cumulative layering, and replicated stream multicasting are given by 91%, 81%, and 73% of the isolated rate. The effective reception rate of replicated stream video multicasting is the largest in Figure 7 (b). Therefore, we can expect that the average video quality of replicated stream video multicasting is the best. The efficiency of replicated stream video multicasting is also the best in Figure 7 (d).

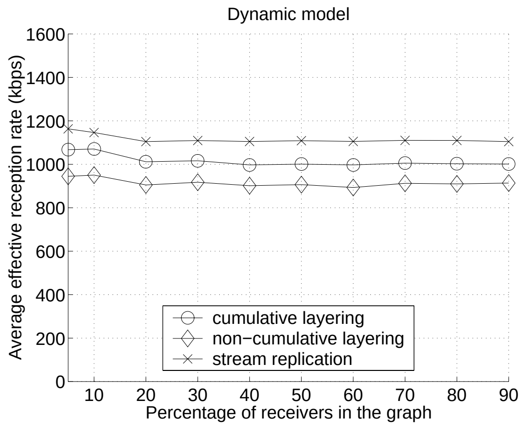
Figure 8 shows the experimental results for the low overhead model. Since no layering overhead is incurred in this model, the average data reception rates in Figure 8 (a) are the same as the average effective reception rates in Figure 8 (b). We expect that the video quality of cumulatively/non-cumulatively layered video multicasting is better than that of replicated stream multicasting, since the average effective reception rate of layered multicasting is larger. The layered multicasting schemes also exhibit more bandwidth consumption and better bandwidth efficiency than replicated stream multicasting.

Figure 6, 7, and 8 show that the performance of each scheme depends on the amount of layering overhead represented by the overhead models. Therefore, to find the conditions under which each scheme is superior, we investigate the effect of the layering overhead and that of the number of layers.

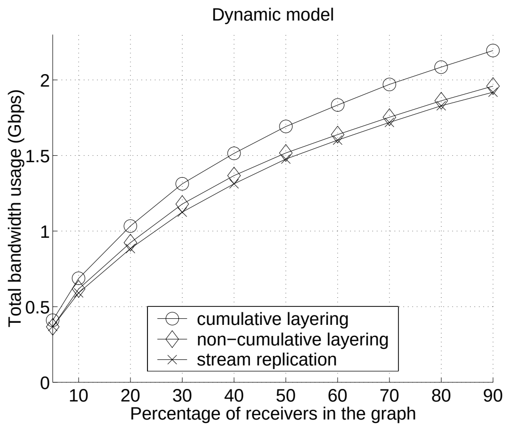
Figure 9 demonstrates the effect of the overhead in layered video multicasting and replicated stream video multicasting in the proportional model. The number of receivers is fixed to 40% of all nodes. Figure 9 (a) shows the effective reception rate of replicated stream video multicasting and layered video multicasting. As layering overhead increases, the effective



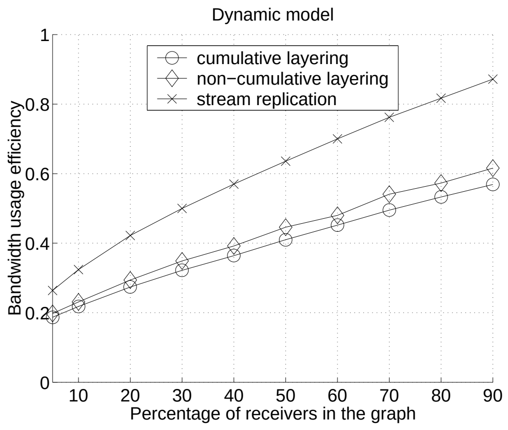
(a) Reception rate



(b) Effective reception rate

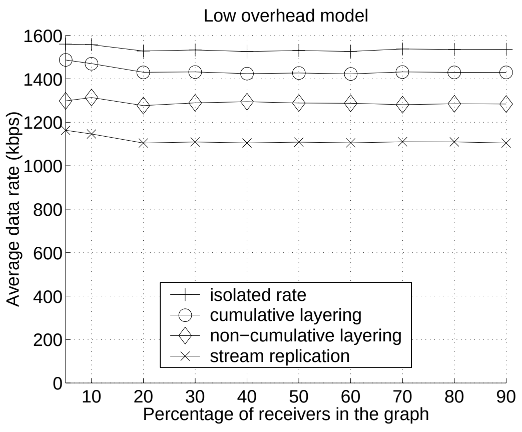


(c) Total bandwidth usage

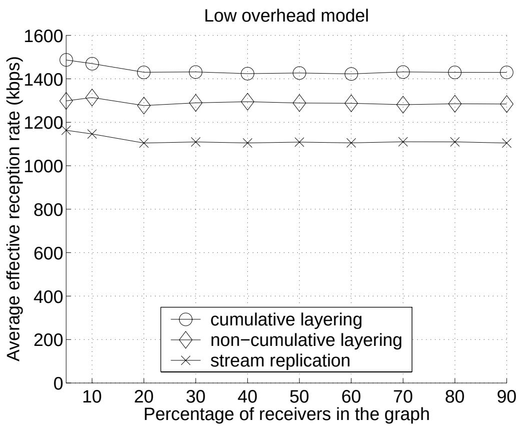


(d) Efficiency

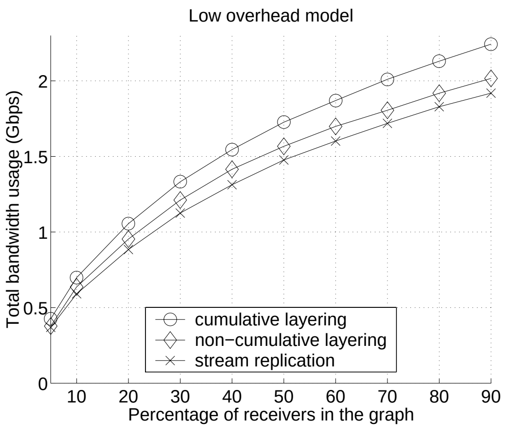
Figure 7: Experiment results in the dynamic model under the random receiver distribution. The results exhibit similar behavior to the proportional model. The average effective reception rate of replicated stream multicasting is the largest in (b).



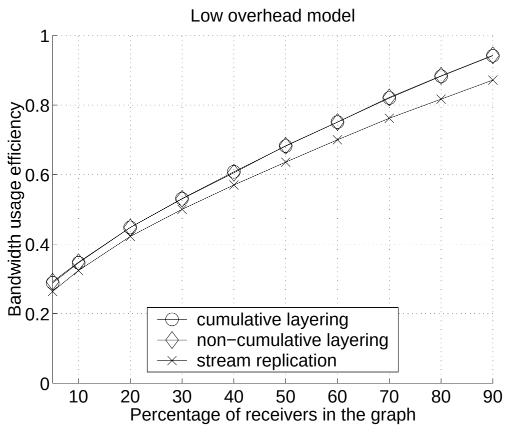
(a) Reception rate



(b) Effective reception rate



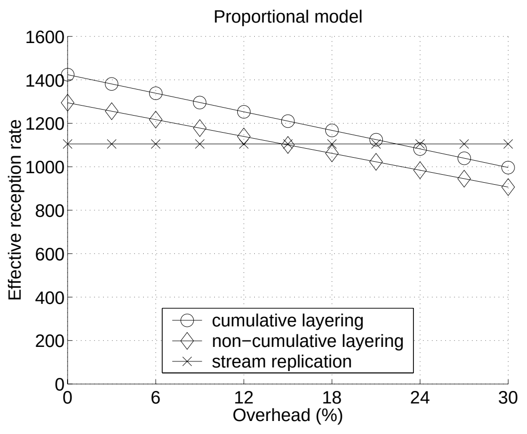
(c) Total bandwidth usage



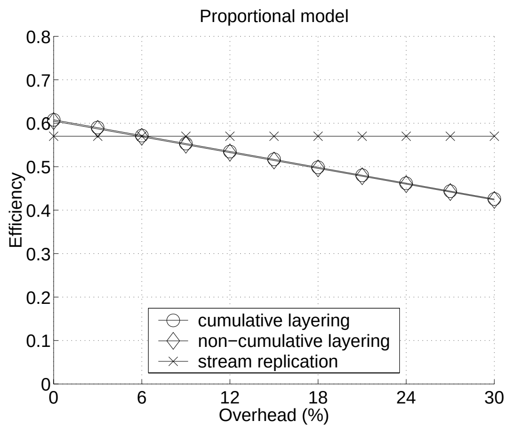
(d) Efficiency

Figure 8: Experiment results for the low overhead model under the random receiver distribution. Since the layering overhead is close to zero, the average data rates and the effective reception rates of layered multicasting are larger than replicated stream multicasting. High bandwidth efficiency of layered multicasting is presented in (d).

reception rate of layered video multicasting is decreased. However, the effective reception rate of replicated stream video multicasting does not change, since no layering overhead is incurred in non-layered streams. The effective reception rate of cumulatively layered multicasting is equal to that of replicated stream video multicasting when the overhead is 22%, and the effective reception rate of replicated stream video multicasting is greater than cumulatively layered video multicasting after that point. The effective reception rate of non-cumulatively layered multicasting is greater than than of replicated stream multicasting when layering overhead is less than 15%. In Figure 9 (b), we investigate how the overhead affects the efficiency. From the experiment, layering overhead of more than 7% tends to favor the replicated stream approach.



(a) Effective reception rate

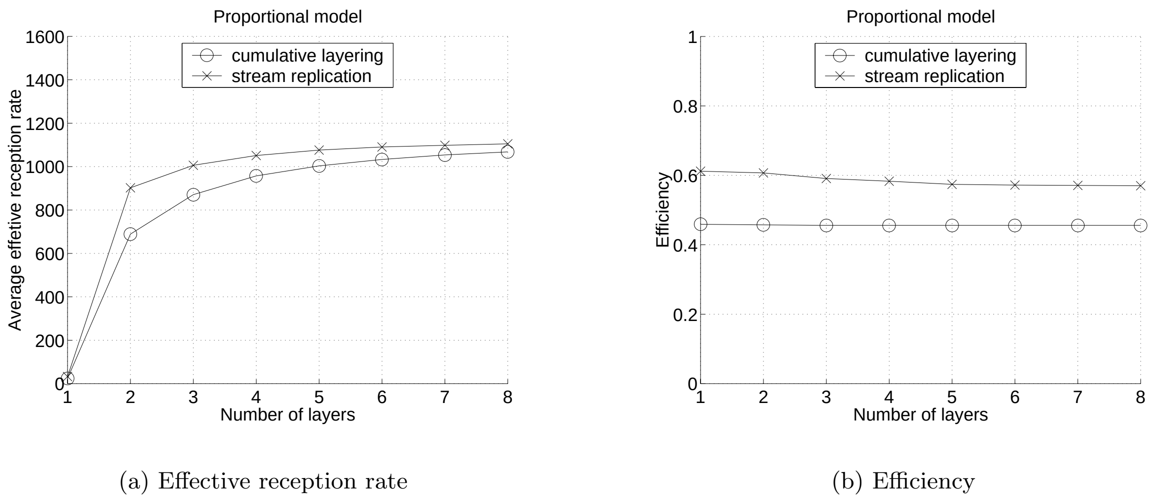


(b) Efficiency

Figure 9: Experiment results for the effect of overhead. (a) shows that the layering overhead of cumulative and non-cumulative layering should be less than 22% and 15% to provide better quality than replicated stream multicasting. Layering approaches also achieve better efficiency when layering overhead is less than 7%.

Based on these observations, it is required that the layering overhead of the cumulative layering should be less than 22% and that of the non-cumulative layering should be less than 15% to provide better video quality. Moreover, to provide both better video quality and higher network efficiency, the layering overhead should be less than 7%.

Figure 10 shows the performance as the number of layers and consequently the number of replicated streams changes in the proportional model. Figure 10 (a) demonstrates that the effective reception rate of replicated stream video multicasting is always greater than that of cumulatively layered video multicasting. The difference gets smaller as the number of layers increases. In Figure 10 (b), the efficiency of stream replication is also greater than that of cumulative layering. From the results, it is expected that the effect of the number of layers on efficiency is not so significant as the amount of layering overhead.



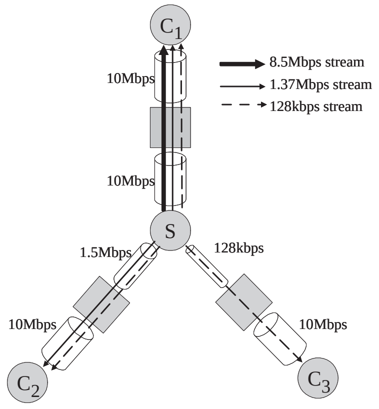
(a) Effective reception rate (b) Efficiency

Figure 10: Experiment results for the effect of the number of layers. When the proportional overhead model is employed, the performance of replicated stream multicasting is always better than cumulative layering.

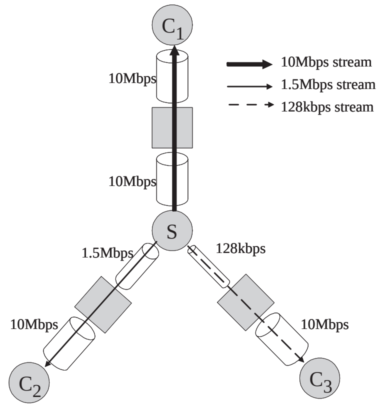
2.5.2 Clustered Distribution

The layered video multicast schemes achieve better bandwidth efficiency when multiple streams share the bottleneck link. When the receivers are placed in one domain, it is more probable that many of the receivers share a bottleneck link. Hence, the layered video multicasting would be more efficient than replicated stream video multicasting.

Consider the example in Figures 11 and 12, which illustrate the impact of network topology on the performance. In Figure 11, the receivers C_1, C_2, and C_3 are placed on different domains and no link is shared by them. We can find that both schemes consume the same amount of bandwidth and the reception rates of C_1, C_2, and C_3 are 10Mbps, 1.5Mbps, and 128kbps. Hence, replicated stream video multicasting has an advantage if we consider layering overhead.



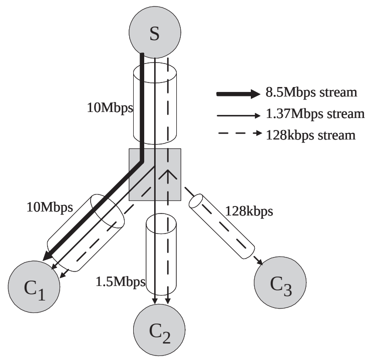
(a) Layered video multicasting



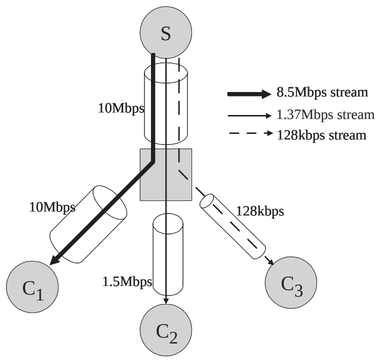
(b) Replicated stream multicasting

Figure 11: An example of random distribution. Since the receivers do not share bottleneck links, replicated stream multicasting exhibits better performance.

On the other hand, Figure 12 presents an example that the receivers C_1, C_2, and C_3 are placed on the same domain and they share the 10Mbps bottleneck link. In this topology, the data reception rates in Figure 12 (a) are larger than that of Figure 12 (b) by 1.63Mbps on average.



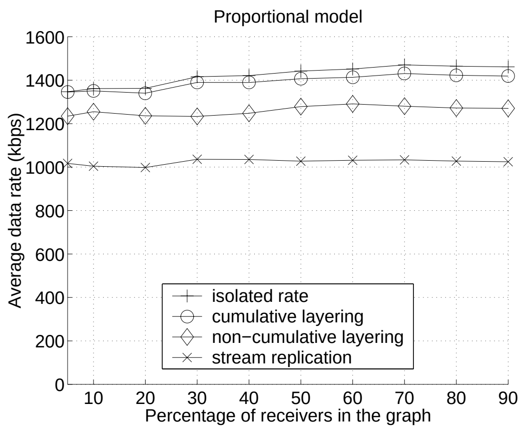
(a) Layered video multicasting



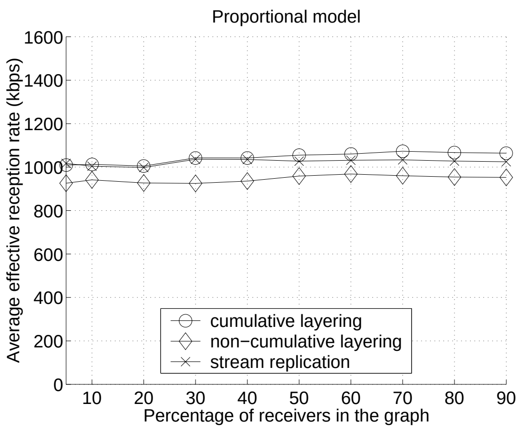
(b) Replicated stream multicasting

Figure 12: An example of clustered distribution. Since the 10Mbps link is shared by all receivers, layered multicasting exhibits better scalability.

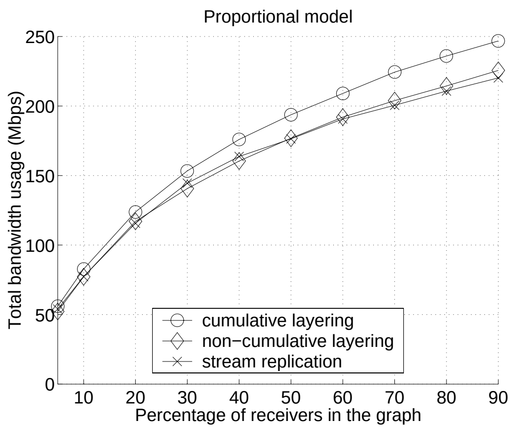
Figure 13, 14, and 15 show the experiment results under the clustered receiver distribution, where receivers are chosen within only one transit domain and a sender is selected from another domain. Compared with Figure 6, 7, and 8, the performance of layered video multicasting is improved but that of replicated stream video multicasting is degraded. Even though some amount of layering overhead is incurred, the effective reception rate of cumulatively layered video multicasting is always greater than that of replicated stream multicasting: 12% in Figure 13 (b), 11% in Figure 14 (b), and 38% in Figure 15 (b). The decrease of replicated stream video multicasting accounts for the decrease of the efficiency. Therefore, we can expect the performance characteristics are changed in favor of layered video multicasting, when receivers are clustered in small number of domains.



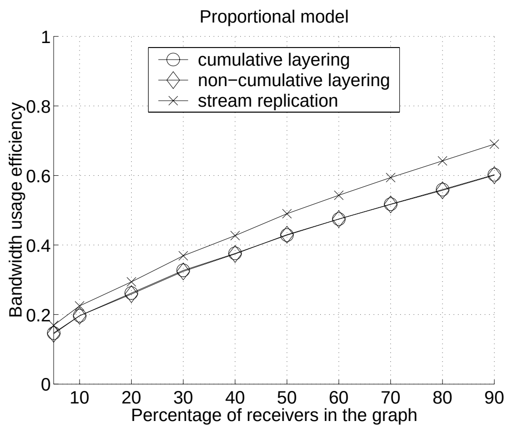
(a) Reception rate



(b) Effective reception rate

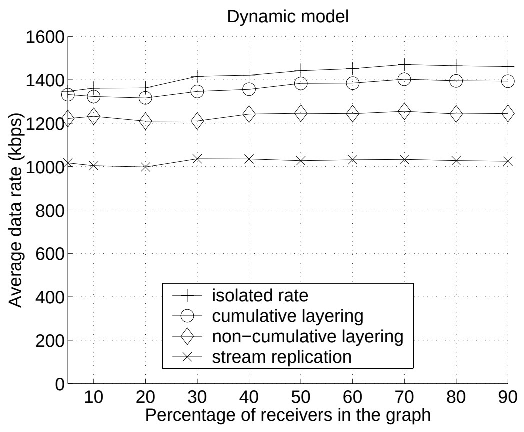


(c) Total bandwidth usage

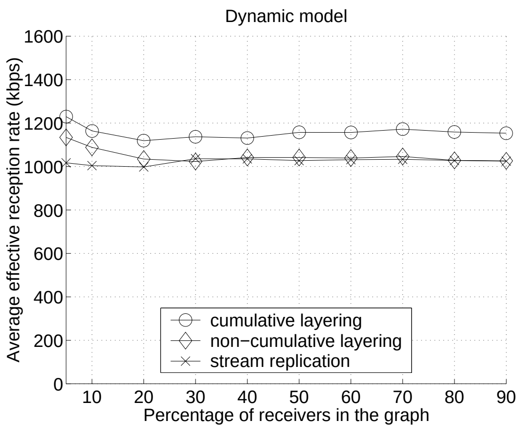


(d) Efficiency

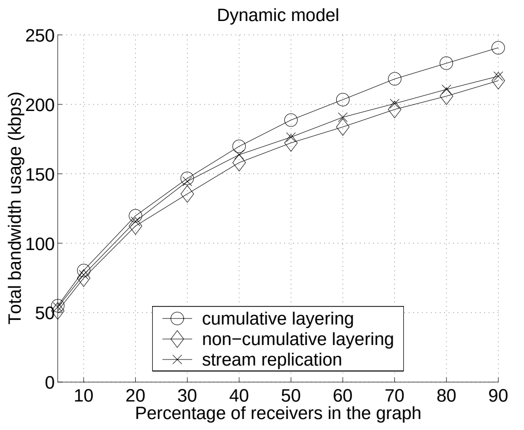
Figure 13: Experiment results in the proportional model under the clustered receiver distribution. Cumulatively layered video multicasting achieves both greater data reception rate and greater effective reception rate which leads to better video quality.



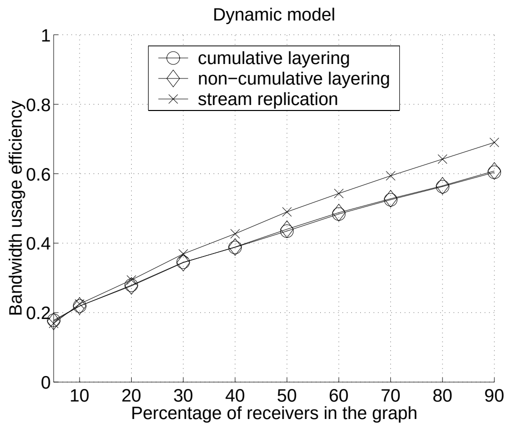
(a) Reception rate



(b) Effective reception rate

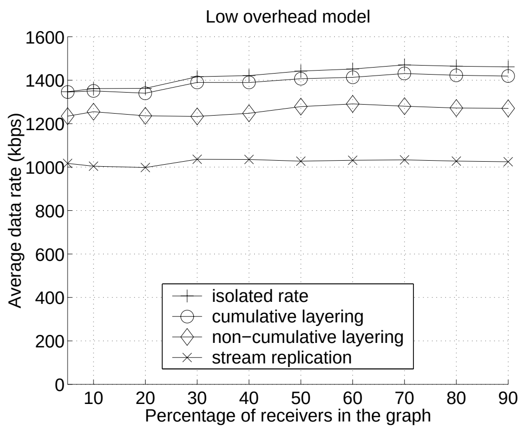


(c) Total bandwidth usage

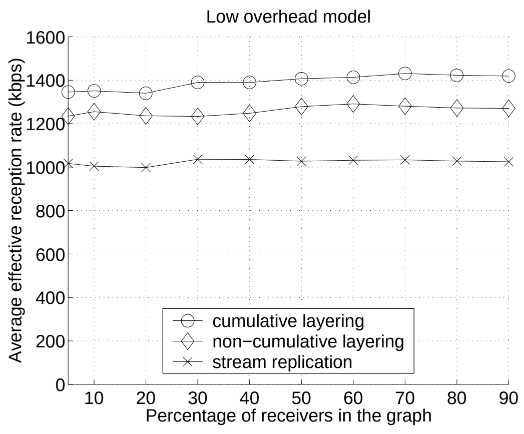


(d) Efficiency

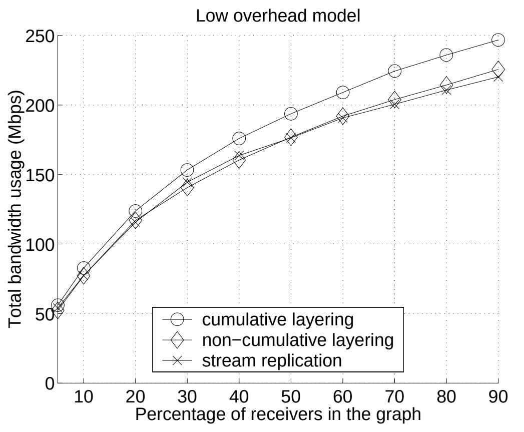
Figure 14: Experiment results in the dynamic model under the clustered receiver distribution. Both cumulatively and non-cumulatively layered video multicasting achieves greater data reception rate and greater effective reception rate than that of replicated stream multicasting.



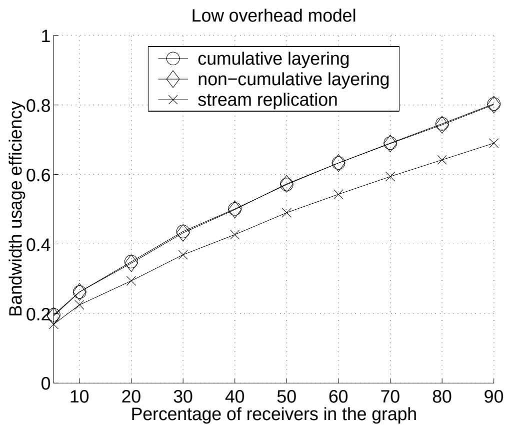
(a) Reception rate



(b) Effective reception rate



(c) Total bandwidth usage



(d) Efficiency

Figure 15: Experiment results for the low overhead model under the clustered receiver distribution. Layered video multicasting schemes exhibit better performance than replicated stream multicasting in all measures.

2.6 Protocol Complexity

In this section, we consider the protocol complexity of cumulatively layered multicasting and replicated stream multicasting, since no existing protocol supports all three schemes. We compare the protocol complexity by using RLM [51]. In RLM, a receiver has the capability to decide whether to drop an additional layer or not. The decision is made by performing a join experiment. Join experiments incur a bandwidth overhead, since a receiver carrying out the experiment sends a join message and multicasts a message identifying the experimental layer to the group. In addition, the shared learning mechanism requires each receiver to maintain significant amount of state information even if it is not necessary.

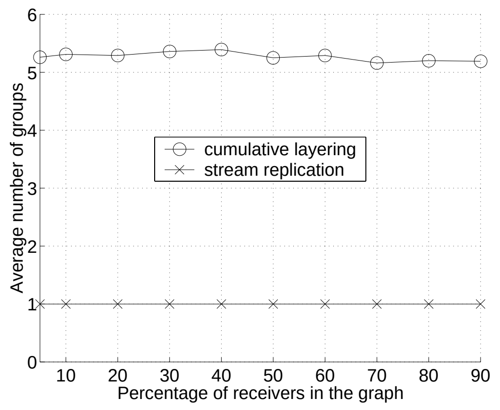
We present below our protocol complexity analysis. The network is modeled by a graph $G = (V, E)$, where V is a set of vertices and E is a set of edges. A set of receivers is defined by $C = \{c_i | c_i \in V, i = 1, ..., n\}$, where n is the number of receivers. A set of video stream is defined by $R = \{r_i | r_i \in \mathbf{R}^+, i = 1, ..., m\}$, where r_i is the data rate of a video stream and m is the number of streams. A set of receivers assigned to stream i is defined by $\Omega_i = \{c_j \mid \phi(c_j) = r_i\}$, where ϕ is the rate allocation function defined by $\phi : C \rightarrow R$. Since we determine the data rate of replicated streams by adding the data rate of the cumulative layering, the receivers in Ω_i can subscribe to the replicated stream β_i, where $\beta_i \leq r_i$, or can accommodate the cumulatively layered stream up to layer i, such as $\alpha_1 + \alpha_2 + \cdots \alpha_i \leq r_i$.

In cumulatively layered video multicasting, the average group size is given by $\frac{1}{m} \sum_{k=1}^{m} k|\Omega_k|$, since a receiver can join multiple groups and $\frac{1}{m}(\sum_{k=1}^{m} |\Omega_k| + \sum_{k=2}^{m} |\Omega_k| + \cdots + \sum_{k=m}^{m} |\Omega_k|) = \frac{1}{m} \sum_{k=1}^{m} k|\Omega_k|$. In a join experiment, cumulatively layered video multicasting requires a receiver send one join message and multicast a message reporting a join experiment to the receivers in the same group. When the link capacity does not change for a long period, the receiver will return to the previous state after a detection time and it has to send a leave message. Hence, the average number of messages in a join experiment is 2 unicast messages and 1 multicast message to $\frac{1}{m} \sum_{k=1}^{m} k|\Omega_k|$ receivers.

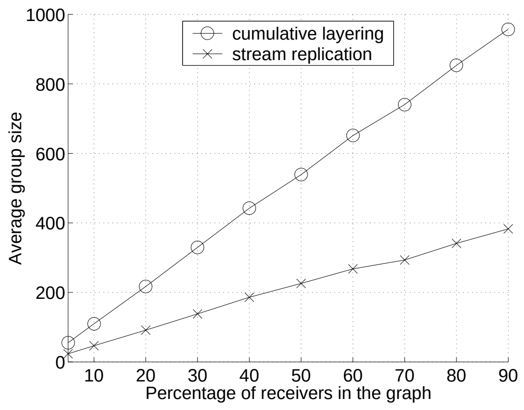
On the other hand, the average group size in replicated stream video multicasting is $\frac{1}{m} \sum_{k=1}^{m} |\Omega_k|$, since every receiver joins only one group. In replicated stream video multicasting, a receiver sends one leave message, one join message, and one multicast message reporting each join experiment. When the link capacity is in the steady state, it has to return to the previous group that involves another one join and one leave messages. The average number of messages in a join experiment is 4 unicast messages and 1 multicast message to $\frac{1}{m}\sum_{k=1}^{m}|\Omega_k|$ receivers.

Therefore, the bandwidth overhead in cumulatively layered multicasting and replicated stream multicasting consists of 2 or 4 unicast messages and 1 multicast message. The cost of a multicast message is dominant when the number of receivers is large.

Figure 16 presents the experiment result of the average group size and the average number of groups joined by a receiver, when the receivers are randomly distributed. In Figure 16 (a), the group size in cumulatively layered video multicasting is twice as large as that in stream replication. Hence, layered video multicasting requires more bandwidth to multicast a message reporting the join experiment and more memory to keep state information. In Figure 16 (b), we can expect that a receiver in a cumulatively layered video multicast session requires more buffer size and better synchronization capability than replicated stream video multicasting, since a receiver in cumulative layering subscribes to more than five layers on average whereas a receiver in stream replication subscribes to only one video stream.



(a) Average group size



(b) Average number of groups

Figure 16: Experiment results for protocol complexity. The nature of joining multiple multicast groups in cumulative layering leads to twice larger group size than replicated stream multicasting. (b) shows that a receiver joins at least five multicast groups.

2.7 Conclusion

In this chapter, we undertake a comparison between the cumulative/non-cumulative layering and replicated stream video multicast schemes. These schemes have been proposed for multicasting to a set of receivers with heterogeneous reception capabilities. While it has been generally accepted that layering is superior to stream replication, this does not appear to be based on a systematic and quantitative comparison of these schemes. We undertake such a comparison here. We first argue that a fair comparison needs to take into account 1) the layering bandwidth overhead, 2) the specifics of the encoding of the layers or replicated streams, 3) the complexity of the protocol required to allow receivers to join and leave the appropriate streams, and 4) the topological placement of receivers relative to each other and relative to the video source. Our results demonstrate the effect of these dimensions on the relative performance of three schemes. They also show the conditions under which each scheme is superior. We summarize several findings from our study:

- The performance of heterogeneous video multicasting schemes depend on the amount of layering overhead. The increase of layering overhead tends to favor replicated stream video multicasting.

- Layering overhead of a scalable video codec should be as small as possible to provide better quality of video. Experiment results show that less than 22% layering overhead is required for cumulatively layered multicasting, and 15% for non-cumulatively layered multicasting to provide better quality than replicated stream multicasting. Both better video quality and higher network efficiency can be achieved when layering overhead is less than 7%. Recall that layering overhead is more than just coding overhead: it also includes the packetization overhead, the protocol overhead, and the error control overhead.

- If the effective reception rate is the same, replicated stream multicasting is preferred. This is because the protocol complexity of replicated stream multicasting is lower than that of layered video multicasting.

- When receivers are clustered in a few domains, layered video multicasting exhibits better performance. One the other hand, when receivers are randomly spread out, the performance of replicated stream video multicasting is better. The receiver distribution can be estimated by `mtrace` [23] when the tracing feature of `mrouted` is implemented in the intermediate multicast routers. If this feature is not supported, application level multicast can be employed [12]. Since an overlay network is generated and optimized in the application level, the topology information is obtained easily. Alternatively, receiver locations can be determined using prior experience from the particular streaming application.

Our work has focused on video multicasting applications. Layering and replication of multicast transmission has also been proposed for bulk-data multicast applications. The layering overhead does not apply in those circumstances, however, the other comparison dimensions will still come into play.

CHAPTER 3

QUALITY ADAPTATION FOR SCALABLE ENCODED VIDEO

3.1 Motivation

In recent years, a number of popular web sites serving multimedia contents have led to the growth of streaming video applications. One of the challenging aspects of carrying video over today's Internet is the fact, as was identified in [56], that the Internet's transmission resources exhibit variability at multiple time-scales, and the available bandwidth fluctuates over a broad range because of the wide distribution of packet loss burst duration, changes in bottleneck capacity, and multiple time-scale queuing-time variation. This dynamic behavior of the Internet makes it difficult to provide perceptually good quality of streaming video.

Small time-scale variability can be accommodated by utilizing a receiver buffer, such that a few video frames are prefetched before they are displayed. However, it is difficult to accommodate large time variability using the receiver buffer because of the buffer size limitation. Large time-scale variability is generally accommodated using scalable (or layered) video encoding [31], [59]. In this approach, a source video is encoded into a base layer and one or more enhancement layers. The base layer can be decoded independently and provides the minimum video quality. Enhancement layers can be decoded cumulatively and contribute to the improvement of video quality.

However, an encoded video can exhibit significant rate variability if the encoding is targeting consistent perceptual quality (this is a result of general compression techniques based on discrete cosine transform (DCT), quantization, motion estimation, motion compensation, and entropy coding). We are, therefore, faced with the problem of trying to accommodate the mismatch caused by both the available bandwidth variability and the encoded video variability.

In this chapter, we investigate quality adaptive video streaming techniques for layered VBR video [40], [41]. The goal is the development of an *optimal* algorithm that minimizes the quality variability while at the same time maximizing the utilization of the variable network bandwidth. Our starting point is traditional rate adaptation in layered video delivery, such that video layers are added and dropped as the available bandwidth changes. However, it is generally agreed that significant quality fluctuation caused by frequent adding and dropping layers may be annoying and degrade the perceptual quality of video.

This chapter is organized as follows. We review related work in Section 3.2. In Section 3.3, we consider quality adaptation algorithms and the question of optimizing quality adaptation. Section 3.4 reports on experimental results we have conducted. This chapter is concluded in Section 3.5. A companion web site provides a demonstration of the basic results of this work [14].

3.2 Related Work

Accommodating data rate variability using the receiver buffer has been widely deployed. Authors in [65] proposed an optimal rate smoothing algorithm based on the work-ahead smoothing technique for non-layered VBR video to achieve minimum variability of transmission rate. Combined with guaranteed service or RCBR service [27], it was shown that network utilization can be increased significantly. However, rate smoothing is not useful for the best effort network, since the Internet does not provide any information about the bandwidth evolution in advance. Also, a smooth data rate does not always guarantee a smooth quality for VBR video.

Coarse grained adaptation of layered video has been discussed to accommodate receiver heterogeneity in the context of multicasting [38], [49], [51]. The number of layers subscribed by a receiver is dynamically varying, since a receiver adjusts video quality based on network condition: subscribing to as many layers as possible when the available bandwidth is large, and dropping layers when the available bandwidth is small. However, frequent adding and dropping of layers can incur significant quality variability which leads to the degradation of perceptual video quality.

Quality adaptation algorithms based on layered video for a unicast environment have been proposed in [18], [53], [61]. The algorithm in [18] transforms the quality adaptation problem into a shortest path problem to minimize variability. The algorithm in [53] tries to maximize perceptual video quality by using bidirectional optimum layer selection. The algorithm in [61] accommodates the short-term rate variability caused by a TCP friendly congestion control mechanism. However, these algorithms assume all layers are CBR encoded, hence they do not maximize perceptual quality of VBR video.

The problem of layered VBR video streaming was first addressed in [66]. The authors modeled the available bandwidth as a stochastic process and proposed an optimal bandwidth allocation scheme between base and enhancement layers. The authors extended this idea and proposed an adaptive heuristic algorithm in [67]. However, as the objective of the scheme is to minimize the loss probability in the base and enhancement layers, it may incur significant quality variation. Since the scheme has the closest (but not the same) objective as we do, we compare the heuristic algorithm with our algorithms and investigate the impact on perceived visual quality.

3.3 Quality Adaptation Algorithms

In this section, we consider the question of quality adaptation for scalable (or layered) encoded video. *Quality adaptation* is defined by a mechanism that adds and drops layers based on the available network bandwidth while maximizing perceptual video quality (primarily we are interested in consistent "long runs" of the same quality video which will be formalized in Section 3.4). We assume that a sender maintains a layered VBR video and that a receiver has some amount of buffering capacity to prefetch an unplayed video. The available bandwidth may exhibit multiple time-scale variability.

3.3.1 Composed Algorithm

As a baseline algorithm, we consider a straightforward approach for quality adaptation by combining existing algorithms. The quality smoothing algorithm proposed in [53] accomplishes the maximum reduction of quality variability for layered CBR video using bidirectional layer selection. To apply this algorithm to layered VBR video, we need to insure that the data rate of the encoded video is sufficiently smoothed to exhibit nearly constant bit rate. We adopt the rate smoothing algorithm presented in [65] for this purpose, since this algorithm enables a sender to transmit a piecewise CBR sequence using the work-ahead smoothing technique. We compose a quality adaptation algorithm from these algorithms.

We divide the receiver buffer into two portions: one used for rate smoothing and the other for quality smoothing. We first use the *FindOptimalSchedule* function in [65] to compute optimally smoothed transmission rate for each layer using rate smoothing buffer. Next, we apply the *MaxAvgRun* function in [53] to the smoothed rate video for maximizing average run length using quality smoothing buffer. The composed algorithm is given in Figure 17, where L is the number of layers and N is the number of video object planes (VOP) in a layer. However, as [53] assumes the size of a frame in a layer is normalized to 1, we need to modify it: replacing 1) $\theta = 1$ by $\theta = x_i[k]$ and 2) $S_k^i = 1$ by $S_k^i = x_i[k]$, where θ is a temporary variable determining a feasible sequence, $x_i[k]$ is the size of VOP k, and S_k^i is a feasible sequence of layer i.

```
1: procedure ComposedAlgorithm ($r$[], $x_i$[], $b_r$, $b_q$)
2:     for $i$ = 1 to $L$ do
3:         FindOptimalSchedule ($x_i$[], $b_r$)
4:     enddo
5:     Initialization: $r_1[k]$ = $r[k]$, where $k \in \{1, \ldots, N\}$
6:     for $i$ = 1 to $L$ do
7:         ($r_{i+1}$[], $s_i$[]) = MaxAvgRun($r_i$[], $x_i$[], $b_q$)
8:     enddo
9: endprocedure
```

Figure 17: Composed algorithm. A piecewise CBR stream is generated by the *FindOptimalSchedule* function, where b_r stands for the rate smoothing buffer size. Next the *MaxAvgRun* function is applied to achieve consistent quality, where b_q specifies the quality smoothing buffer size.

3.3.2 Optimal Quality Adaptation

3.3.2.1 Framework

In formulating optimal quality adaptation, we consider a discrete-time model. Let $x_i[k]$ be the VOP size of the ith layer at time k, b_i be the receiver buffer size for storing unplayed ith layer video, and $r_i[k]$ be the available bandwidth, where $i = 1, \ldots, L$ and $k = 1, \ldots, N$. It is assumed that the size of $x_i[k]$ is variable but known in advance which is generally true in the transmission of stored video. Also note that the available bandwidth $r_i[k]$ is the residual bandwidth after accommodating layers $1, 2, \ldots, i-1$. Hence the condition of $r_i[k] > 0$ implies $r_j[k] > 0$, where $j = 1, \ldots, i-1$. Conversely $r_i[k] = 0$ implies $r_{j'}[k] = 0$, where $j' = i+1, \ldots, L$. $X_i[k]$ represents the cumulative data requirement in the ith layer defined by $X_i[k] = \sum_{j=1}^{k} x_i[j]$. The cumulative capacity, $C_i[k]$, is defined to quantify transmission resources. The cumulative capacity of the ith layer is determined by two constraints: the receiver buffer size and the available network bandwidth, such that

$$C_i[k] = \min(S_i[k-1] + b_i,\ C_i[k-1] + r_i[k]),$$

where $S_i[k]$ is the cumulative selected data defined by $\sum_{j=1}^{k} s_i[j]$, and $s_i[j]$ is the selected data defined by

$$s_i[j] = \begin{cases} x_i[j], & \text{if VOP } j \text{ is selected for transmission} \\ 0, & \text{otherwise.} \end{cases}$$

Note that a transmission scheduling algorithm may exploit $s_i[k]$ to deliver the ith layer: if $s_i[k]$ is not zero, a VOP of the ith layer at time k is transmitted to a receiver. On the other hand, if $s_i[k]$ is zero, it is not transmitted to the receiver. Especially, $s_i[k]$ is said to be *feasible* if the amount of cumulative selected data does not exceed cumulative capacity, i.e., $S_i[k] = \sum_{j=1}^{k} s_i[j] \leq C_i[k]$. Note that the feasibility implies the existence of a scheduling algorithm to transmit the selected data, but it does not provide a specific scheduling algorithm. The contribution of this research is the development of quality adaptation algorithms that determine which VOPs are to be transmitted in each layer to maximize perceived video quality rather than the investigation of specific scheduling algorithms.

Figure 18 illustrates the framework of optimal quality adaptation. We assume that there are enough transmission resources in the beginning stage, such that a sender can transmit the ith layer without discontinuity until k_0, and therefore $S_i[k] = X_i[k]$, $k = 1, \ldots, k_0 - 1$. At time k_0, a receiver stops displaying layer i because of the feasibility, and the sender determines a future time k_1 at which the receiver will resume displaying layer i. We choose k_1 to be the point that satisfies $C_i[k_1] = S_i[k_1] + d$ (we show how d can be determined in Section 3.3.2.4). Note that after k_0 there is still some available capacity for layer i.

The sender allows the receiver to prefetch future VOPs of the ith layer starting from k_1. During (k_1, k_2), the receiver starts displaying buffered video using this capacity and also keeps prefetching VOPs for layer i. At k_2 where $S_i[k]$ meets $C_i[k]$ again, the receiver stops displaying the ith layer and begins to prefetch future layer i VOPs.

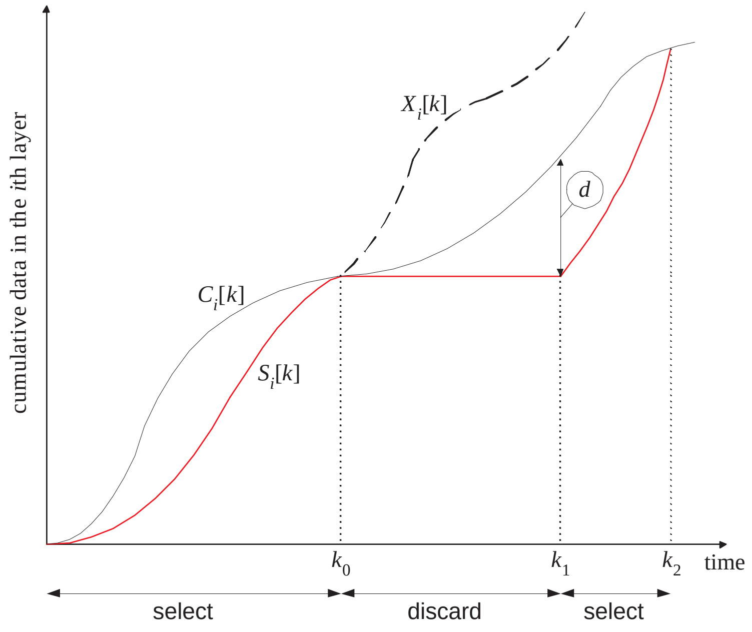


Figure 18: Framework of quality adaptation. A receiver makes transitions between the `select` and `discard` state based on transmission resources.

Although the quality adaptation framework in Figure 18 seems similar to the rate adaptation mechanism in [65], there is a significant difference: the constraint of rate adaptation is determined by the receiver buffer size and the source video rate, whereas the main constraint of quality adaptation is transmission resources.

We model the quality adaptation mechanism in the ith layer by a two-state machine as shown in Figure 19. In each layer, a VOP in the `select` state will be transmitted, and one in the `discard` state dropped. Each state is determined by the amount of the cumulative capacity and the VOP size. Since we cannot send more data than can be accommodated by transmission resources, the cumulative capacity is an upper bound of the cumulative transmission schedule of the ith layer: a sender transmits the ith layer without discontinuity as long as the cumulative capacity is greater than the cumulative selected data.

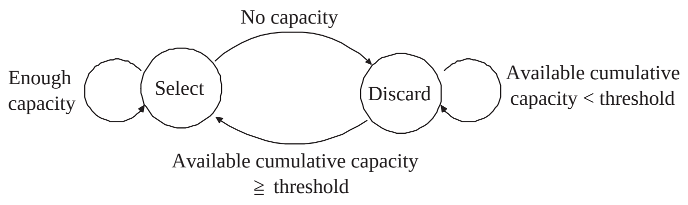

No capacity
Enough capacity
Select
Discard
Available cumulative capacity < threshold
Available cumulative capacity ≧ threshold

Figure 19: State transition diagram describing quality adaptation framework. A state transition takes place based on the amount of capacity and threshold.

3.3.2.2 Select State

In the ith layer, a VOP in the `select` state will be transmitted to a receiver. To achieve maximum perceptual quality, it is necessary to reduce quality variability, such that a sender transmits as many consecutive VOPs as possible. Hence, if the sender enters the `select` state, it does not leave the state until the amount of the cumulative capacity is not greater than the cumulative selected data. Once the cumulative capacity is not enough to accommodate the cumulative selected data, the sender enters the `discard` state.

To accommodate multiple layers, we employ a conservative transmission policy: if there are available transmission resources, layered video is transmitted as soon as possible from the base layer to the highest enhancement layer. The residual cumulative capacity of lower layers can be used to accommodate higher layers. In this scenario, lower layers are protected better than higher layers, since lower layers are buffered before higher layers. Therefore, if a VOP of the ith layer is selected, VOPs of layer $1, 2, \ldots, i-1$ are also selected. Conversely, if a VOP of the ith layer is not selected, VOPs of layer $i+1, i+2, \ldots, L$ cannot be selected.

Sometimes a startup latency is needed to prefetch a few initial VOPs. Otherwise, it is likely to discard the beginning part of the video unless the initial network bandwidth is large. The startup latency is accomplished by shifting the encoded video by σ, such that

$$x_i[k] = \begin{cases} 0, & 0 < k \leq \sigma \\ x'_i[k-\sigma], & k > \sigma, \end{cases}$$

where $x'_i[k-\sigma]$ is the VOP size of original encoded video in the ith layer.

3.3.2.3 Discard State

In the `discard` state, a receiver stops displaying layer i as much as possible to reduce the quality variability. However, as dropping many VOPs leads to the network under-utilization, a sender has to return to the `select` state if a certain condition is satisfied. In Figure 19, the sender leaves the `discard` state when the available cumulative capacity exceeds a threshold. The *available cumulative capacity* is defined by the difference of the cumulative capacity and the cumulative selected data (e.g., the available cumulative capacity exceeds a threshold d at k_1 in Figure 18, and the sender leaves the `discard` state at this point). The question here is how to determine the threshold. Theorem 1 in Section 3.3.2.4 states that a threshold equal to the receiver buffer size achieves both consistent video quality and the necessary condition of the maximum network utilization.

Note that the selected data $s_i[k]$ is zero in the `discard` state, since no VOP of the ith layer is selected for transmission. On the other hand, the available cumulative capacity is not zero although it is smaller than the threshold. We can take advantage of the available capacity to prefetch future VOPs of the ith layer.

The optimal quality adaptation algorithm is shown in Figure 20. This is an implementation of the state machine in Figure 19 with the threshold value of the receiver buffer size. Note that this algorithm assumes the available network bandwidth information is known *a priori*. When there is a transition from the `select` state to the `discard` state, it is necessary to determine the next prefetch point. Line 16 implies that the point can be determined, only when the available network bandwidth information is available.

In optimal quality adaptation, lower layers will exhibit less quality fluctuation and better bandwidth utilization than higher layers, since quality adaptation is applied from the base layer to the highest layer. The performance of the algorithm depends on both the receiver buffer size b_i and the available network bandwidth $r_i[k]$, since the cumulative capacity is determined by the receiver buffer size and the available bandwidth. The *UpdateBandwidth* function computes the residual available bandwidth for higher layers.

```
procedure OptimalAdaptation ($r[]$, $x_i[]$, $b_i$)
    Initialization: $r_1[k] = r[k]$, $S_i[k]$ = 0, $C_i[k]$ = 0
        where $i \in \{1, \ldots, L\}$ and $k \in \{1, \ldots, N\}$
    for $i$ = 1 to $L$ do
        for $k$ = 1 to $N$ do
            $C_i[k] = \min(S_i[k-1] + b_i,\ C_i[k-1] + r_i[k])$
            if $s_i[k-1]$ = $x_i[k-1]$ then
                if $C_i[k] \geq S_i[k-1] + x_i[k]$ then
                    $s_i[k]$ = $x_i[k]$
                    $S_i[k] = S_i[k-1] + x_i[k]$
                else
                    $s_i[k]$ = 0
                    $S_i[k] = S_i[k-1]$
                endif
            else
                if $C_i[k] \geq S_i[k-1]$ + $b_i$ then
                    $s_i[k]$ = $x_i[k]$
                    $S_i[k] = S_i[k-1] + x_i[k]$
                else
                    $s_i[k]$ = 0
                    $S_i[k] = S_i[k-1]$
                endif
            endif
        enddo
        UpdateBandwidth($r[]$, $C_i[]$, $S_i[]$)
    enddo
endprocedure
```

Figure 20: Optimal quality adaptation algorithm. This is an implementation of the state transition diagram with the threshold of receiver buffer size.

3.3.2.4 Optimality of Algorithm

The optimality of quality adaptation is defined by the minimum quality variability while maximizing network utilization. To achieve this goal, an optimal adaptation framework is proposed in Section 3.3.2.1. Based on this framework, we need to maximize the sojourn time in the `select` state and the `discard` state for minimum quality variability. Theorem 1 states that a threshold value equal to the receiver buffer size achieves maximally consistent video quality under the constraint of transmission resources utilization. Theorem 1 also implies that the quality adaptation algorithm should exploit transmission resources as much as possible to achieve both minimum quality variation and the necessary condition of maximum network utilization.

Theorem 1 *In the framework of the optimal quality adaptation, a threshold value equal to the* receiver buffer size *satisfies 1) minimum video quality variability and 2) the necessary condition of maximum network utilization.*

Proof: see Appendix B.

3.3.3 Realtime Adaptation

The optimal quality adaptation algorithm in Section 3.3.2 assumes the available bandwidth information is known in advance. Since the Internet does not provide any information about the bandwidth evolution, we need an algorithm that minimizes quality variability without using future bandwidth information. In this section, we develop a realtime adaptation algorithm to satisfy this requirement.

The realtime adaptation algorithm is shown in Figure 21. The framework of the realtime adaptation is similar to the optimal adaptation algorithm. The differences between the realtime adaptation and the optimal adaptation are 1) the realtime adaptation makes a decision on which layer and which VOP to be transmitted in real time (lines 4-6), and 2) a sender makes a receiver prefetch future layer i VOPs when there is a transition from the `select` state to the `discard` state (line 15).

The question is how to determine the prefetch point at the transition time. We consider

```
procedure RealtimeAdaptation ($r[]$, $x_i[]$, $b$)
    Initialization: $Y_i[k]$ = 0, $C_i[k]$ = 0, $b_i[k]$ = $b$,
        where $i \in \{1, \ldots, L\}$ and $k \in \{1, \ldots, N\}$
    for $k$ = 1 to $N$ do
        $\hat{r}[k]$ = SmoothBandwidth($r[k]$)
        for $i$ = 1 to $L$ do
            $C_i[k] = \min(Y_i[k-1] + b_i[k],\ C_i[k-1] + r[k])$
            if $s_i[k-1]$ = SELECT then
                if $C_i[k] \geq Y_i[k-1] + x_i[k]$ then
                    $s_i[k]$ = SELECT
                    $Y_i[k] = Y_i[k-1] + x_i[k]$
                else
                    $s_i[k]$ = DISCARD
                    $Y_i[k] = Y_i[k-1]$
                    NextSelect = EstimateNextSelect($\hat{r}$[k])
                endif
            else
                if $k \geq$ NextSelect then
                    $s_i[k]$ = SELECT
                    $Y_i[k] = Y_i[k-1] + x_i[k]$
                else
                    $s_i[k]$ = DISCARD
                    $Y_i[k] = Y_i[k-1]$
                endif
            endif
            UpdateBandwidth($r[k]$, $C_i[k]$, $Y_i[k]$)
        enddo
    enddo
endprocedure
```

Figure 21: Realtime adaptation algorithm. This algorithm performs quality adaptation without using the future bandwidth information.

a moving average (MA) estimator to determine the prefetch point. The MA estimator is simple and widely known for the TCP retransmission timeout estimation in [33]. Using the MA estimator, the available bandwidth can be estimated by the weighted sum of smoothed average (sr) and the mean deviation (d). The estimated available bandwidth ($\hat{r}$) is given by the following relationship and is implemented in the *SmoothBandwidth* function:

$$\begin{aligned} err &\leftarrow r[k] - sr \\ sr &\leftarrow sr + 0.125err \\ d &\leftarrow d + 0.25(|err| - d) \\ \hat{r} &\leftarrow sr + 4d \end{aligned}$$

The prefetch point of layer i is determined by $\min([\frac{b_i}{\hat{r}}], M)$ in the *EstimateNextSelect* function, where M is the maximum duration to the prefetch point. This relationship is a linear estimation of the prefetch point, such that the threshold will be equal to the receiver buffer size. We limit the duration by M to prevent estimation errors from causing under-utilization of transmission resources.

It should be noted that the performance of the realtime adaptation algorithm depends on the available bandwidth estimator. Therefore, we need to ensure that the estimation should be carried out effectively. The author in [56] investigated the characteristics of the Internet delay variation. The measurement results show that the variation ranges primarily on time-scales of $0.1 - 1$ seconds, although frequently the time-scales can be much larger. The authors in [78] investigated the stationarity of the Internet. One of the findings in the research is that the stationarity of Internet path properties depends on time-scales. Experimental results show that the stationarity of packet loss rate is well preserved on time-scales of a few seconds to minutes. Since the throughput of a transport protocol is mainly determined by delay and loss rate [25], [55], we can expect that the throughput stationarity is maintained on time-scales of a few seconds to minutes. The results in [78] show that we can expect a few minutes of stationarity to prevent available bandwidth from varying by more than $\pm10\%$. Therefore, we can expect reasonable performance of the bandwidth estimator by limiting the region of operating points (i.e., by setting the maximum duration M to a few seconds or a few minutes).

3.4 Evaluation

In this section, we show results from experiments by which we evaluate our algorithms (the offline optimal algorithm and realtime adaptation) and compare them to the baseline *composed* algorithm. The main goal is to evaluate the performance of the algorithms and the impact of parameters, such as the receiver buffer size or transport protocols, on perceptual video quality. The experimental results including encoded video traces, network simulation script, and decoded video are available at the companion web site [14].

3.4.1 Rate Variability

In this section, we investigate the rate variability of scalable encoded video using the MPEG-4 FGS codec. MPEG-4 FGS provides an efficient scalable video coding and is highly adaptable to the bandwidth fluctuation by distributing the enhancement layer data over a wide range of bit rates [47], [59].

The MPEG-4 FGS structure consists of a base layer, and one or two enhancement layers. Two types of enhancement layers are defined in hybrid temporal-SNR scalability: 1) *SNR FGS layer* contributes to enhancing video quality by adding DCT coefficients with a reduced quantization step size which leads to highly accurate DCT coefficients and high quality video. 2) *Temporal FGS layer* is designed to improve temporal resolution by providing a higher frame rate.

Figure 22 shows the data rate of an encoded 175-second scene from the *A river runs through it* movie. The video sequence is generated by a software MPEG-4 FGS codec in [59]. Specific encoding parameters are as follows: CIF resolution, 24 frames/sec, GOP size being 12, and the number between anchor frames being 3. The average data rates are 160kbps in the base layer, 582kbps in the FGS layer, and 1.27Mbps in the FGST layer, hence the overall average data rate is approximately 2.0Mbps. It should be noted that MPEG-4 FGS incurs bandwidth overhead as much as 672kbps compared with non-scalable coding (i.e., the data rate of a single layer encoded video is 1.33Mbps at the same quality). This is because non-scalable coding is optimized at a specific data rate, whereas MPEG-4 FGS is optimized over a data rate *range* [47]. Authors in [38] investigated the effect of the overhead in detail by comparing layering and stream replication in video multicasting.

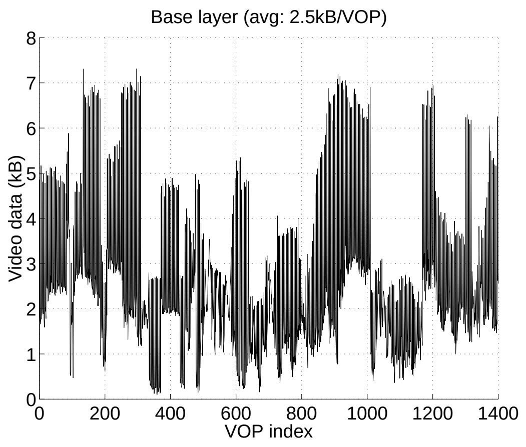
Figure 22 shows all layers exhibit significant rate variability. The variability spans 7.4kbytes in the base layer, 16.5kbytes in the FGS layer, and 16.4kbytes in the FGST layer.

To achieve optimal layering, we need to investigate the relative importance of the two types of enhancement layer. Figure 23 shows two possible implementations of the hybrid temporal-SNR scalability structure. The structure in Figure 23 (a) places emphasis on the FGS layer to improve video quality, while the structure in Figure 23 (b) increases temporal resolution before improving video quality. Since there is a tradeoff between the picture quality and temporal resolution, a sender has to choose an appropriate structure based on the characteristics of a video. Typically, a video with slow motion favors the structure in Figure 23 (a), whereas a video with high motion does the structure in Figure 23 (b). In this research, the structure in Figure 23 (a) is employed, since the degree of the motion of the movie is relatively low (i.e., the base layer is considered as layer 1, the FGS layer as layer 2, and the FGST layer as layer 3).

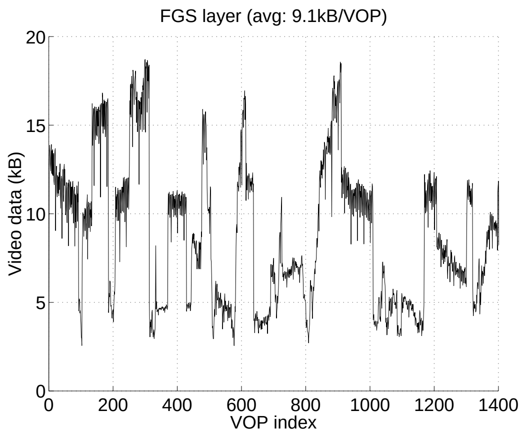
3.4.2 Bandwidth Variability

We generate bandwidth variability resulting from each underlying transport protocol over the single bottleneck topology: senders are arranged on one end of a link, and receivers on the other end. All links except the bottleneck link have sufficient capacity so that any packet drop occurs at the bottleneck link. The bottleneck link capacity is set to 3Mbps and link delay to 20ms, whereas the access links have 100Mbps and 2ms delay. We run `ns-2` [54] experiments over the network model. To model bandwidth variability, we consider the throughput experienced between a TFRC sender and a TFRC receiver, or between a TCP sender and a TCP receiver. TFRC throughput is measured by counting the number of packets from the receiver-side router to the TFRC receiver, and TCP throughput is measured between the receiver-side router to the TCP receiver. To simulate a realistic background traffic, a 700kbps UDP flow is generated by superposing 100 ON/OFF sources of pseudo nodes which have the Pareto distribution [75].

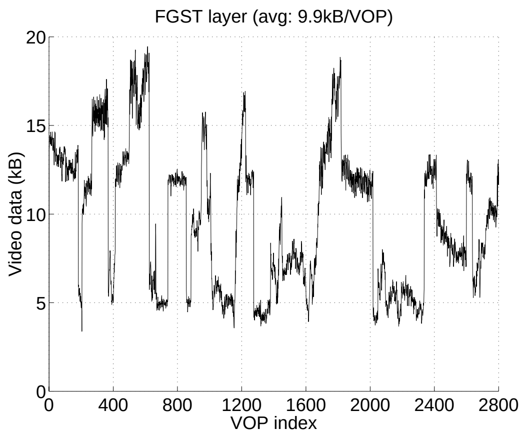
We consider the use of our algorithms over both TCP and TCP friendly rate control



(a) Base layer

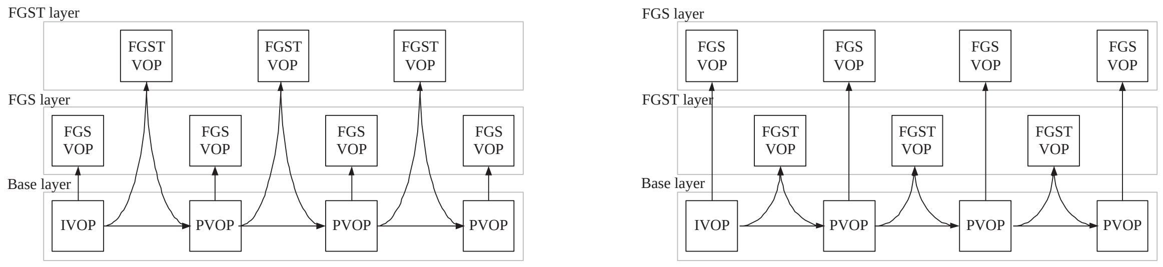


(b) FGS layer



(c) FGST layer

Figure 22: Data rate variability of a scalable encoded video. The video sequence is generated from *A river runs through it* using an MPEG-4 FGS codec.



(a) FGS-FGST

(b) FGST-FGS

Figure 23: MPEG-4 FGS hybrid scalability structures. Structure (a) improves video quality first and structure (b) increases temporal resolution first.

(TFRC). Since TCP is the dominant protocol in the Internet, it is reasonable to employ TCP for video streaming. However, TCP has been regarded as inappropriate for video streaming, since the nature of additive increase multiplicative decrease incurs significant data rate variability. TFRC was proposed to overcome the disadvantage of TCP and provide a congestion control mechanism with TCP fairness and smoothly-changing data rate that leads to reduced bandwidth variability [25].

To accommodate received packets, we need to allocate buffer space to each layer. Based on the relative data rates, we assign 10% of the space to the base layer, 30% to the FGS layer, and 60% to the FGST layer respectively[1]: (e.g., if a receiver is equipped with 100kbytes buffer, allocated buffer sizes for base, FGS, and FGST layers are 10kbytes, 30kbytes, and 60kbytes each).

3.4.3 TFRC Performance with a Long Duration Background Traffic

We first compare the performance of quality adaptation algorithms when a TFRC flow competes with a TCP and a Pareto flows. It is assumed that the characteristic of the background traffic does not change during the experiment.

Figure 24 shows the bandwidth variability of TFRC, where the throughput is averaged over 150ms intervals. Although TFRC is slowly responsive to packet loss that achieves reduced variability of available bandwidth, the sluggishness of TFRC leads to an unnecessary

[1]Note that this provides slightly more buffering for base layer than would be proposed in [66]. We found this to be necessary to protect against losses resulting from burstiness of the base layer.

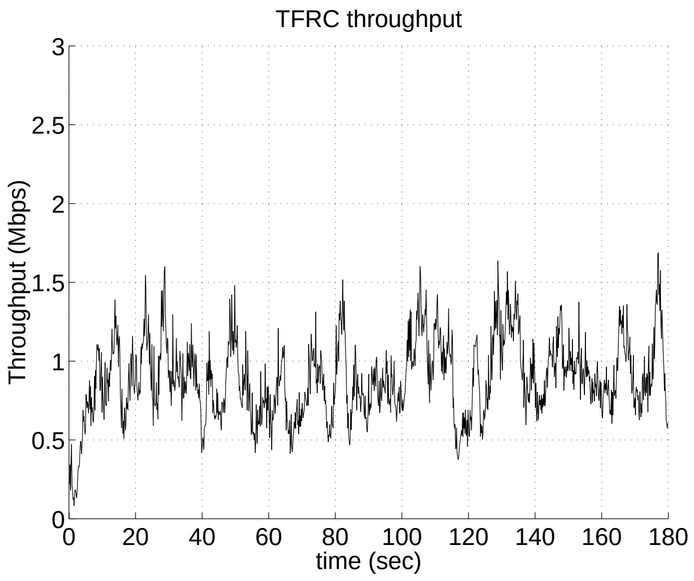


Figure 24: TFRC throughput with a long duration background traffic. TFRC exhibits small rate variability, slow response time, slightly less throughput compared with TCP.

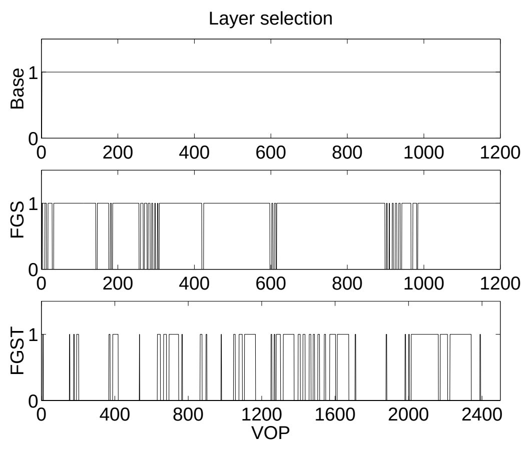
decrease in throughput and an increase in response time under the dynamic environment [4]. Figure 24 demonstrates that the steady state throughput is 930kbps, slightly lower than the fair share, and that the response time to reach the steady state is 5 seconds.

In Figure 25 and 26, we present experimental results of the composed algorithm and the optimal adaptation algorithm in terms of the layer selection function, $\phi(s_i[k])$, which is defined by $\phi(s_i[k]) = 0$ if $s_i[k] = 0$, and $\phi(s_i[k]) = 1$ otherwise (i.e., 0 stands for discarded VOPs and 1 for selected VOPs). Note that both algorithms assume a knowledge of available bandwidth information. In the composed algorithm, we allocate half of the receiver buffer to the rate smoothing buffer and the remaining half to the quality smoothing buffer.

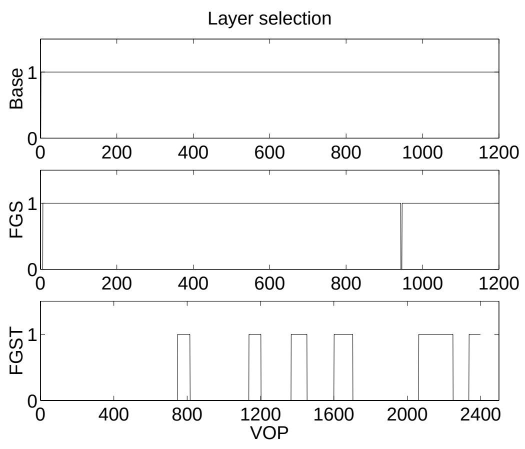
Figure 25 shows that the composed algorithm can reduce the quality variability by increasing the receiver buffer size. The quality transition in the FGST layer is 74 for 100kbytes buffer, but it is reduced to 11 for 1Mbytes buffer, where the *quality transition* of the ith layer is defined by $\sum_{k=1}^{N} I_i(k)$ and

$$I_i(k) = \begin{cases} 1, & \text{if } \phi(s_i[k-1]) \neq \phi(s_i[k]) \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Figure 26 shows the results of the optimal quality adaptation algorithm. Compared with Figure 25, the results show that the quality variability is significantly decreased and the

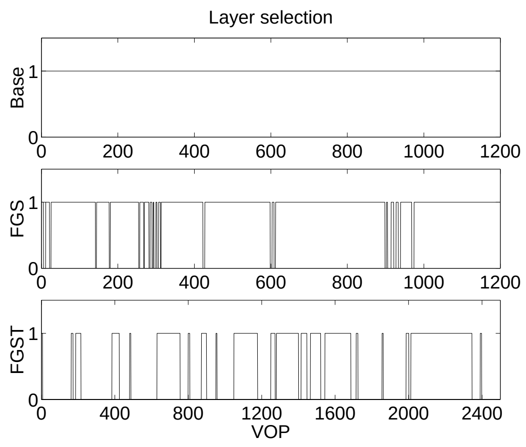


(a) 100kbytes buffer

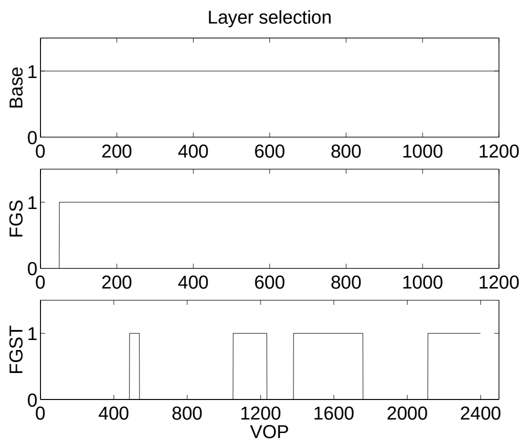


(b) 1Mbytes buffer

Figure 25: Performance of the composed algorithm over TFRC. Quality transitions are reduced as the receiver buffer size is increased.



(a) 100kbytes buffer



(b) 1Mbytes buffer

Figure 26: Performance of the optimal adaptation algorithm over TFRC. Quality transitions are reduced as the receiver buffer size is increased and smaller than that of the composed algorithm

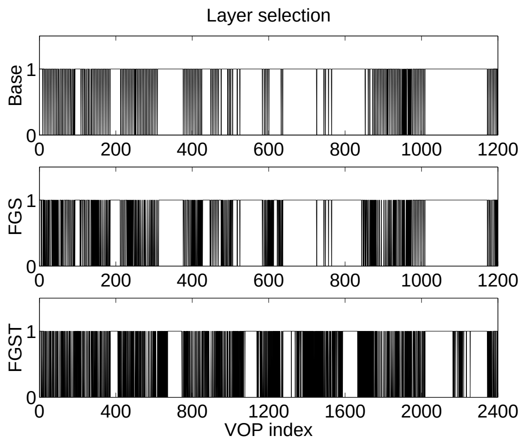
length of run is increased, where a *run* is defined by a sequence of consecutive VOPs. The quality transition can also be decreased when we increase the receiver buffer size: there are 39 transitions in the FGST layer for 100kbytes buffer, but transitions are reduced to 7 for 1Mbytes buffer. Note that a lot of initial VOPs in enhancement layers are dropped because of the slow response time of TFRC. Although we set the startup latency to 3 seconds to prefetch a few initial VOPs, it turns out to be not enough to accommodate enhancement layers, since the average network throughput is significantly lower than the aggregated data rate of encoded video.

We also compare the realtime adaptation with the threshold-based streaming algorithms in [66] and [67] (note that these algorithms do not require a knowledge of future bandwidth). The authors in [66] first proposed a static threshold policy to minimize loss probability in the base and the enhancement layers. They extended the idea and proposed an algorithm in [67] that adjusts threshold values dynamically. We extend the original algorithms to three layers as shown in Appendix C.

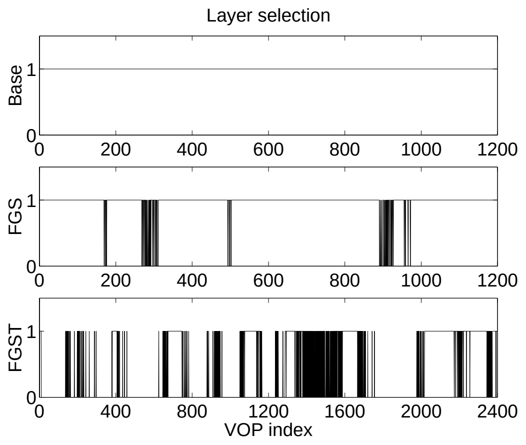
Figure 27 and 28 show that threshold-based streaming algorithms exhibit a lot of quality transitions and a small average run. When the buffer size is small, the static threshold algorithm in Figure 27 incurs significant quality fluctuations. When the buffer size is increased, both threshold-based algorithms show similar performance. Note that the performance improvement of the dynamic threshold algorithm is marginal, even though the buffer size is increased. This is because the threshold values that determine relative resource allocation are adjusted by the data rate of encoded video and the measured bandwidth throughput, but not affected by the buffer size.

Figure 29 presents the performance of the realtime adaptation. Experimental results show 83 quality transitions in the FGST layer for 100kbytes buffer, however it is reduced to 22 for 1Mbytes buffer. Compared with Figure 27 and 28, the performance of realtime adaptation is better than thresold-based streaming. This is because the objective of the threshold policy was to minimize loss probability, not to maximize perceptual quality.

We investigate the performance for other video sequences as well: *A river runs through it*, *Jurassic park I*, and *Starwars: episode I*. We measure the performance in terms of

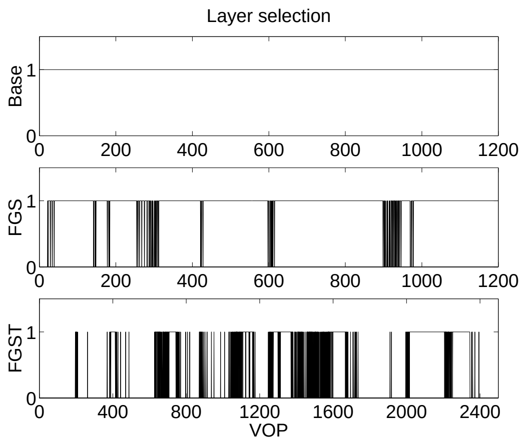


(a) 100kbytes buffer

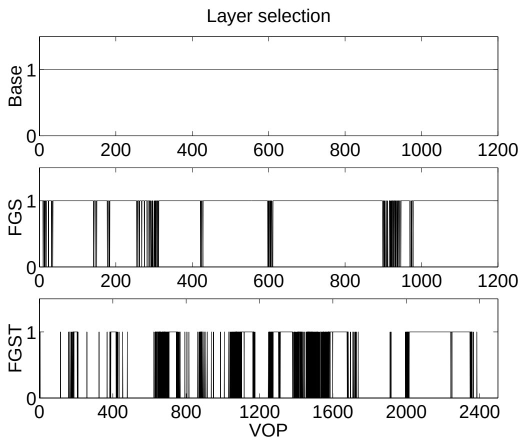


(b) 1Mbytes buffer

Figure 27: Performance of the static threshold streaming over TFRC. Large number of quality transitions are observed for small buffer size.

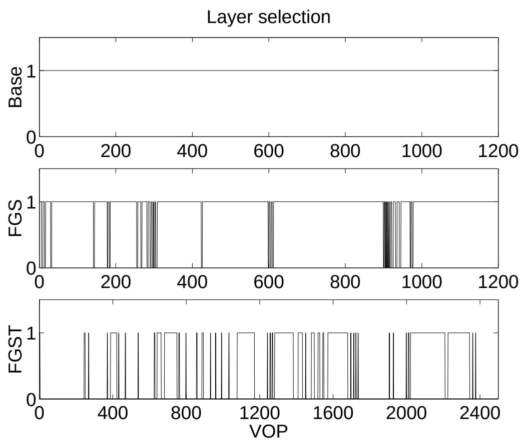


(a) 100kbytes buffer

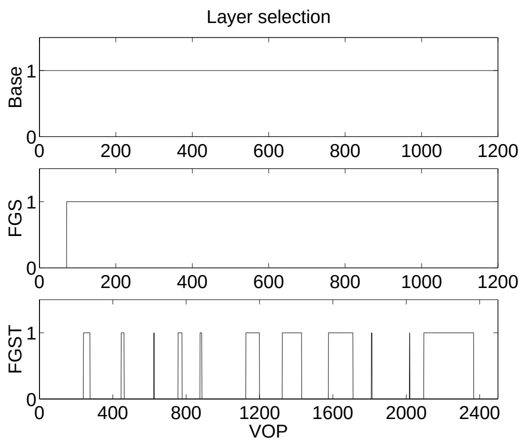


(b) 1Mbytes buffer

Figure 28: Performance of the dynamic threshold streaming over TFRC. (b) shows that the performance improvement is marginal in spite of the buffer size increase.



(a) 100kbytes buffer



(b) 1Mbytes buffer

Figure 29: Performance of the realtime adaptation over TFRC. Quality transitions are reduced as the receiver buffer size is increased. The performance is comparable to the composed algorithm and the optimal adaptation algorithm.

the weighted average quality transition (WAQT) and the weighted average run length (WARL) [40]:

$$\begin{aligned} WAQT &= \textstyle\sum_{i=1}^{L} \sum_{k=1}^{N} w_i I_i(k) / \sum_{i=1}^{L} w_i \\ WARL &= \textstyle\sum_{i=1}^{L} \sum_{j=1}^{k_i} \frac{w_i n_j}{k_i} / \sum_{i=1}^{L} w_i, \end{aligned}$$

where w_i is a relative weight for each layer, $I_i(k)$ is defined in (1), k_i is the number of runs in the ith layer, and n_j is the length of the jth run. Since lower layers are more important than higher layers, w_i is defined by $w_1 = 0.6$, $w_2 = 0.3$, and $w_3 = 0.1$. Note that the weights are determined in an inversely proportional manner with respect to the allocated buffer size in Section 3.4.2.

Table 1 shows the performance of all algorithms. In all cases, the optimal adaptation algorithm exhibits the smallest WAQT, since the algorithm is optimized to minimize quality variability. Moreover, the optimal adaptation algorithm shows the largest WARL. Hence, we can expect the optimal adaptation generates the longest video with the smallest quality transitions. The results also show that the performance of the realtime adaptation is not as good as optimal adaptation, but it is comparable to that of the composed algorithm. This is not surprising since both the optimal and composed algorithms operate with a knowledge of future bandwidth variability.

Table 1: Experiment results for 3 video sequences (THR: dynamic threshold streaming, RT: realtime adaptation, COMP: composed algorithm, OPT: optimal adaptation algorithm)

	Buffer size	100kbytes			
	Algorithm	THR	RT	COMP	OPT
WAQT	*A river runs through it*	80.9	25.1	23.3	**15.3**
	Jurassic park	47.5	7.2	7.0	**2.0**
	Starwars: episode I	107.1	8.0	7.2	**1.6**
WARL	*A river runs through it*	1451	1464	1465	**1478**
	Jurassic park	1799	2164	2163	**2179**
	Starwars: episode I	60	2163	2163	**2183**

	Buffer size	1Mbytes			
	Algorithm	THR	RT	COMP	OPT
WAQT	*A river runs through it*	79.0	2.5	2.6	**1.0**
	Jurassic park	40.1	1.0	1.6	**0.4**
	Starwars: episode I	62.5	1.0	1.6	**0.6**
WARL	*A river runs through it*	1450	2122	1804	**2152**
	Jurassic park	2160	2165	2206	**2355**
	Starwars: episode I	107	2168	2206	**2237**

3.4.4 TFRC Performance with a Dynamic Background Traffic

We now investigate the performance of quality adaptation algorithms when a competing flow exhibits significant dynamic behavior. To model dynamic network condition, we add an ON/OFF CBR flow that starts at 50 seconds and stops at 100 seconds. When it is active, the CBR source generates a 600kbps traffic. Figure 30 shows the experimental result of the TFRC throughput. Observe that the responsiveness of the TFRC protocol makes the throughput decreased during the active period. When the CBR source goes idle, the TFRC throughput is increased again.

Performances of quality adaptation algorithms with 1Mbytes buffer are compared in Figure 31. Figure 31 (a) and (b) show the performance of the composed algorithm and the optimal adaptation algorithm. Compared with the long duration background scenario in Section 3.4.3, it is observed that the performances of quality adaptation algorithms are degraded during the congestion period (e.g., the composed algorithm and the optimal adaptation algorithm as well as the realtime adaptation show 4 more quality transitions in the FGS layer than Section 3.4.3 during the VOP index of [400, 800]). This is due

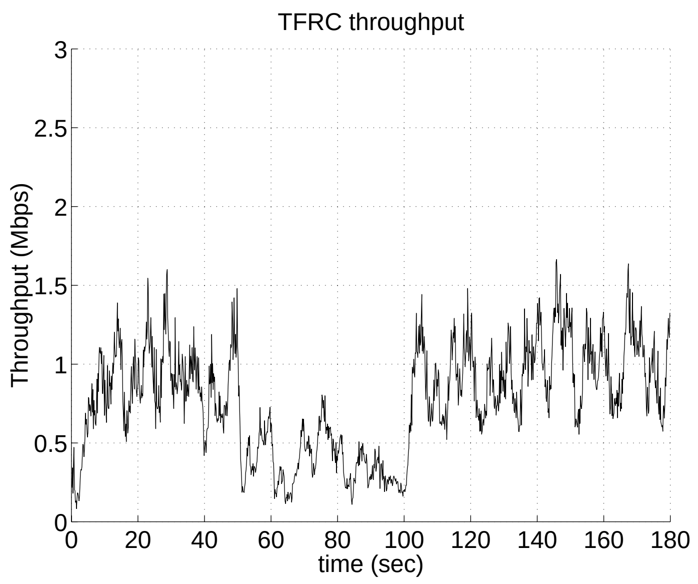


Figure 30: TFRC throughput under a dynamic background traffic. A competing flow is active during [50, 100] seconds.

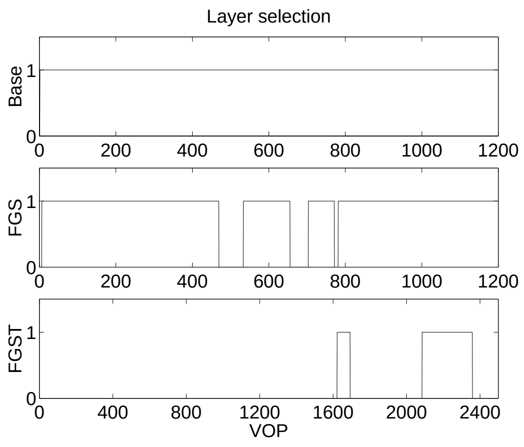
to the decrease of TFRC throughput to achieve the fair-share of network bandwidth: the throughput decrease leads to the decrease of cumulative capacity which makes a sender leave the `select` state. Note that base layer still does not show any quality variability, since lower layer is accommodated before higher layers.

Figure 31 (c) and (d) show the performance of the dynamic threshold streaming and the realtime adaptation. Figure 31 (c) shows that the dynamic threshold streaming exhibits a lot of quality transitions in both FGS and FGST layers. However, the realtime adaptation in Figure 31 (d) shows significantly less quality transitions in both layers.

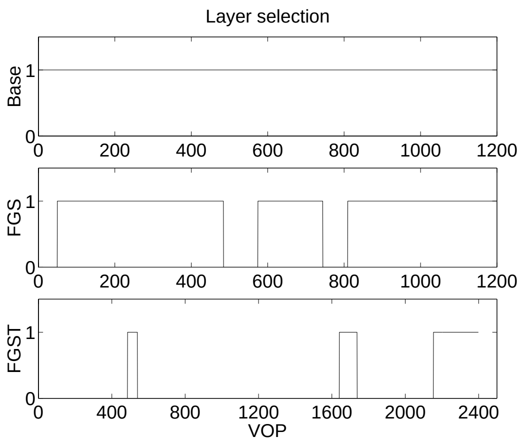
Note that the available bandwidth estimator in Section 3.3.3 gets benefit from the TFRC protocol: since the slow change of the TFRC throughput will reduce the estimated mean deviation, the estimated available bandwidth does not vary much, and therefore the estimation error will be reduced.

3.4.5 TCP Performance with a Long Duration Background Traffic

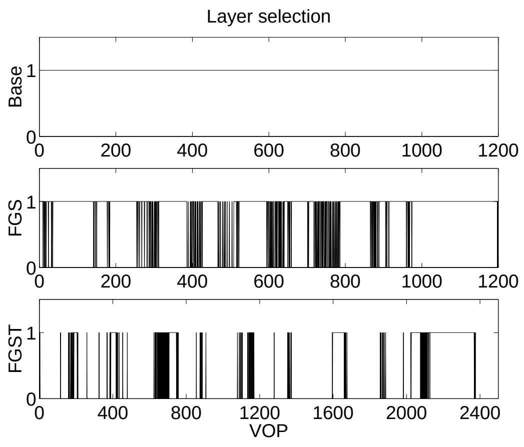
In this section, we compare the performance of quality adaptation algorithms over TCP. Figure 32 shows the bandwidth variability of TCP, when there is a long duration background traffic. As reported in [4], a TCP flow achieves more throughput than a TFRC flow



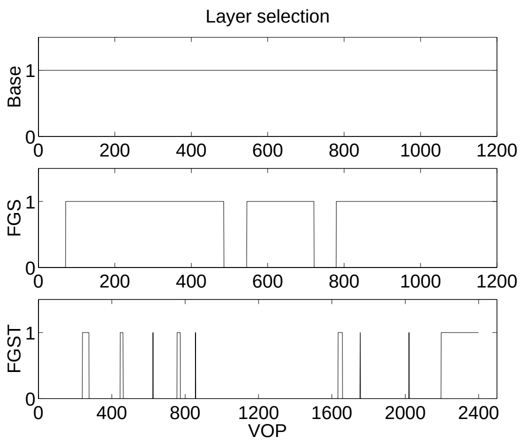
(a) Composed algorithm



(b) Optimal adaptation



(c) Dynamic threshold algorithm



(d) Realtime adaptation

Figure 31: Performance comparison over TFRC with 1Mbytes buffer. Performance of quality adaptation algorithms are degraded during the congestion period.

in dynamic network conditions, since the fast responsiveness of TCP leads to an aggressive behavior. Experimental result shows that the average throughput of the TCP flow is 1.1Mbps, 15% greater than that of the TFRC flow in Section 3.4.3. However, high sensitivity to packet losses results in significant small time-scale variability as much as 1.5Mbps even in the steady state.

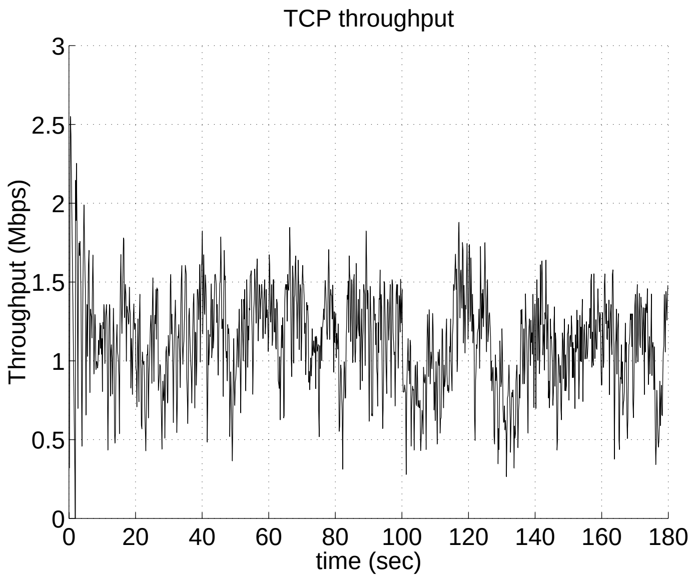
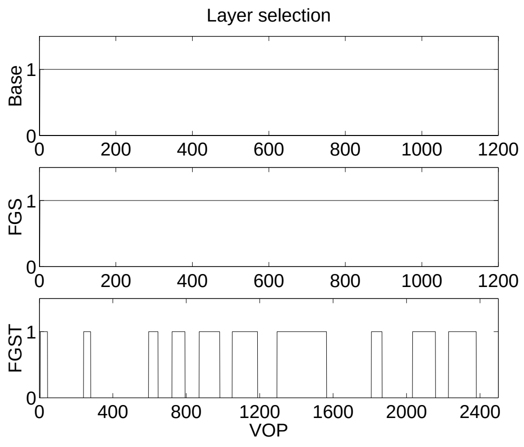
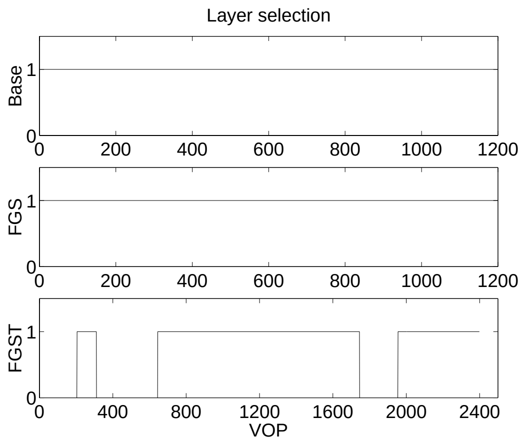


Figure 32: TCP throughput under a long duration background traffic. TCP exhibits fast response time, more throughput, and large rate variability.

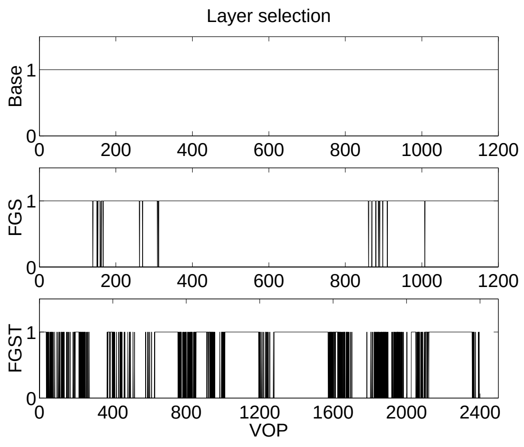
Figure 33 (a) and (b) show the performance of the composed algorithm and the optimal adaptation algorithm when running over TCP. Compared with Figure 25 and 26, we can find that the performance is significantly improved especially in the transient state, such that the composed algorithm does not exhibit any quality transitions in the FGS layer. We can also find that quality transitions are reduced and the length of run is increased. Note that the performance in the steady state is also better than TFRC. For example, if we consider the steady state performance in the FGST layer in optimal adaptation, the number of quality transitions in Figure 33 (b) is smaller than Figure 26 (b), and the average run length of TCP is longer than TFRC. Two reasons account for the superiority of TCP: 1) TCP achieves more throughput than TFRC in dynamic conditions, 2) although TCP exhibits significant small time-scale variability, it can be successfully accommodated by the receiver buffer.



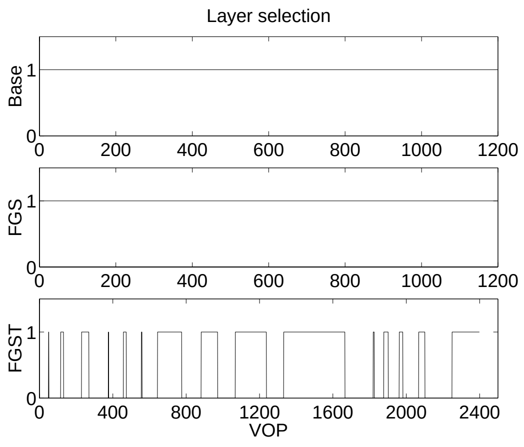
(a) Composed algorithm



(b) Optimal adaptation



(c) Dynamic threshold algorithm



(d) Realtime adaptation

Figure 33: Performance comparison over TCP with 1Mbytes buffer. Overall performance is better than TFRC, since TCP achieves more throughput in dynamic conditions, and small time-scale variability can be accommodated by the receiver buffer.

Figure 33 (c) and (d) show the performance of the dynamic threshold streaming and the realtime adaptation that do not require prior knowledge of the available bandwidth information. The results show that the dynamic threshold streaming exhibits a lot of quality transitions, even though 1Mbytes receiver buffer is used. However, the realtime adaptation in Figure 33 (d) does not show any quality transitions in both the base and the FGS layers. Compared with Figure 33 (a), we can find that the realtime adaptation shows comparable performance to the composed algorithm.

3.4.6 TCP Performance with a Dynamic Background Traffic

We also investigate the performance over TCP when a dynamic background traffic exists. Figure 34 shows TCP throughput from the network simulation. We use this throughput as a model of bandwidth variability. As reported in [4], a TCP flow achieves more throughput than a TFRC flow in dynamic network conditions. Fast responsiveness of TCP leads to an aggressive behavior before it reaches the steady state. Even in the steady state, TCP achieves slightly more throughput than TFRC. However, small time-scale variability is significant as much as about 3Mbps in the steady state, since TCP is very sensitive to packet losses.

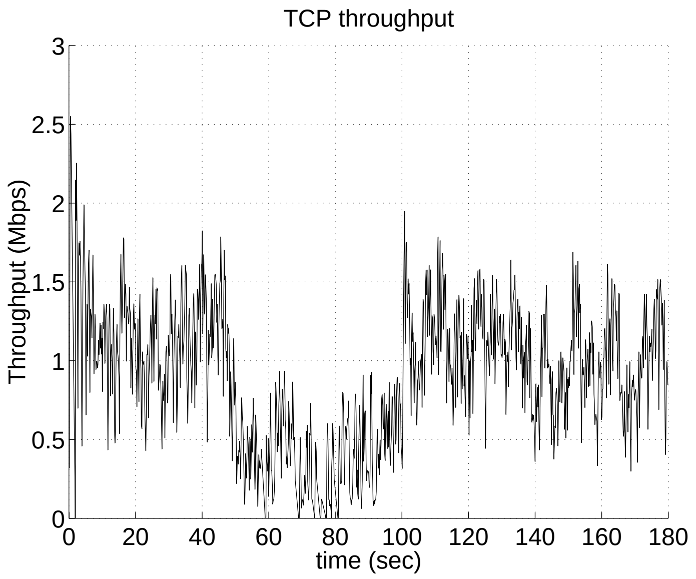


Figure 34: TCP throughput under a long duration background traffic. A competing flow is active during [50, 100] seconds.

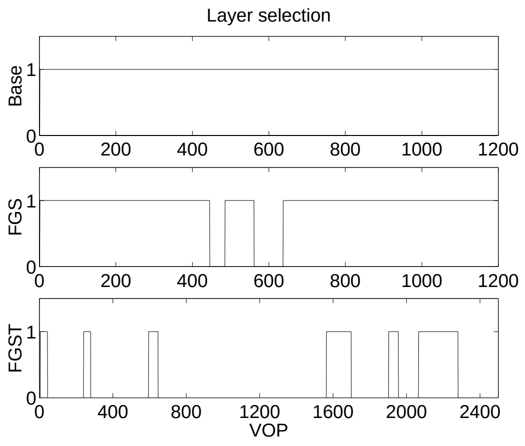
Figures 35 (a) and (b) show the performance of the composed algorithm and the optimal adaptation algorithm when running over TCP. Compared with Figure 31, we can find that the performance is significantly improved, especially in the transient state. Note that the performance in the steady state is still better than TFRC. For example, if we consider the FGST layer after 400 VOPs in optimal adaptation, the number of quality transitions in Figure 31 (b) is equal to Figure 35 (b) but the average run length is longer than TFRC.

Figure 35 (c) and (d) show the performance of the dynamic threshold algorithm and the realtime adaptation which do not require prior knowledge of the available bandwidth information. The results show that the dynamic threshold algorithm exhibits a lot of quality transitions. On the other hand, in Figure 35 (d), the realtime adaptation shows significantly less quality transitions in both the FGS and the FGST layers. Compared with Figure 31, we can find that the realtime adaptation shows similar performance to the composed algorithm.

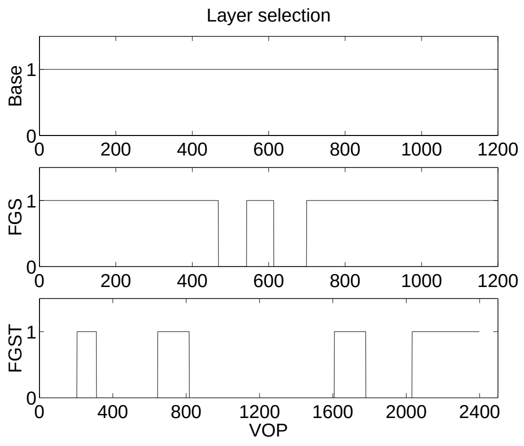
In summary, it is observed that TCP throughput exhibits both small time-scale and large time-scale variability, and the throughput is decreased when the background traffic is active. However, since small time-scale variability can be accommodated by the receiver buffer, all quality adaptation algorithms reduce quality variability significantly. To accommodate large time-scale variability caused by the active background traffic, a few FGS layer VOPs are dropped during the VOP index of [400, 800]. However, the base layer does not show any quality variability, and the overall performance is comparable to Figure 31.

3.5 Conclusion

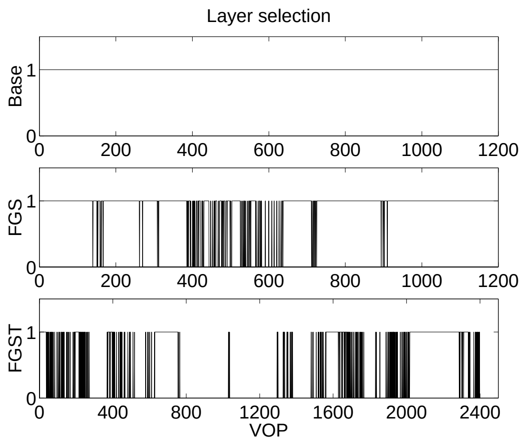
In this chapter, we consider the problem of providing perceptually good quality of a scalable encoded VBR video. The problem is challenging because both transmission resources and encoded video exhibit multiple time-scale variability. To accommodate the variability, we develop an optimal adaptation algorithm that minimizes quality variability while increasing the usage of the available bandwidth. We prove the optimality that a transition threshold should be equal to the receiver buffer size using the concept of cumulative capacity. We then propose a realtime adaptation algorithm that does not require a knowledge of future bandwidth variability. Experimental results demonstrate that quality adaptation



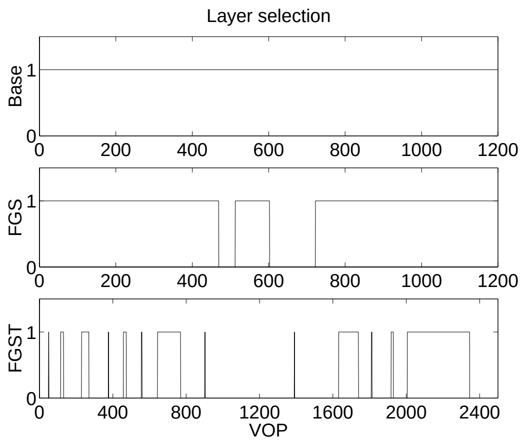
(a) Composed algorithm



(b) Optimal adaptation



(c) Dynamic threshold algorithm



(d) Realtime adaptation

Figure 35: Performance comparison over TCP with 1Mbytes buffer. To accommodate large time-scale variability caused by the network congestion, a few FGS layer VOPs are dropped.

algorithms reduce the quality variability significantly and that the optimal quality adaptation algorithm exhibits the best performance. We also show that our quality adaptation provides better results when running over TCP as compared with TFRC. This is because TCP achieves higher throughput, and small time-scale variability of TCP can be accommodated by receiver buffer. Because perceptual quality is hard to assess by graphs and numbers alone, we have provided videos on a companion web site [14] which demonstrates the perceptual differences among the different algorithms and options.

CHAPTER 4

VIDEO STREAMING OVER TCP

4.1 Motivation

Since TCP is the dominant protocol in the Internet, it is reasonable to employ TCP for video streaming: recent measurement study in [74] has reported that 44% of video streaming flows are actually delivered over TCP. Especially, in many situations, video streaming servers are located behind firewalls that permit only pre-specified port numbers. In this scenario, video streaming over TCP is the only choice to get around the firewalls using well-known port numbers (e.g., HTTP or RTSP). Also, the reliable packet delivery of TCP is important, when error resilience is not implemented in a video codec.

While the use of TCP provides reliable video stream delivery, it is difficult to provide good quality of streaming video over TCP: 1) the sawtooth behavior of additive increase and multiplicative decrease (AIMD) incurs significant data rate variability, and 2) the use of retransmission timeouts may introduce unacceptable end-to-end delay, and the retransmitted data may be delivered too late for display.

These drawbacks of TCP can be mitigated to some extent through the use of receiver-side buffering (e.g., see [18], [45], [66]). The buffer size has to be large enough to insure that video data is not lost. In current practice, however, there are no guidelines for the provisioning of the receiver buffer, and smooth playout is insured through *over-provisioning.*

We are interested in memory-constrained devices (e.g., mobile phones or PDAs) where it is desirable to determine the *right* playout buffer size. This chapter, therefore, considers the question of how large the playout buffer should be in order to achieve desired performances for a video streaming over TCP. To this end, we characterize video streaming over TCP in a systematic and quantitative manner [42]. Our starting point is an analytic model of a video streaming system. Based on this model, we quantify buffer size requirements of three scenarios: 1) when TCP throughput matches video encoding rate, 2) when TCP throughput is smaller than the encoding rate, and 3) when TCP throughput is limited by the maximum window size. Both simulation results and the Internet experiments validate our model and demonstrate the minimum buffering delay can achieve desired quality with high accuracy.

The remainder of the chapter is organized as follows. In Section 4.2, we present a video streaming model and derive buffer size requirements. Section 4.3 shows simulation results to validate our model. Section 4.4 reports on the Internet experiment results we have conducted. Related work is described in Section 4.5. This chapter is concluded in Section 4.6.

4.2 Model and Analysis

In this section, we present a video streaming model over TCP. Based on the model, we derive buffer size requirements for three different scenarios. 1) when TCP throughput matches video encoding rate, 2) when TCP throughput is smaller than the encoding rate, and 3) when TCP throughput is limited by the maximum window size.

4.2.1 Video Streaming Model over TCP

Figure 36 shows a video streaming model consisting of a sender and a receiver. We assume that the sender transmits CBR video packets over a unicast TCP connection, and that the receiver is equipped with a playout buffer in front of a video decoder. The decoder waits to fill the buffer before displaying video. There are two types of buffering delay:

- *Initial buffering delay*: to accommodate initial throughput variability or inter-packet jitters, it is needed to employ initial buffering delay. While a streaming application achieves more tolerance with larger initial buffering, it increases the startup latency and response time.

- *Rebuffering delay*: the decrease of TCP throughput might cause a playout buffer to be empty. When this happens, a decoder stops displaying video until it receives enough video packets. Note that rebufferings take place in the middle of a session, and therefore buffering delay requirements for a long video stream are determined by the congestion avoidance algorithm of TCP.

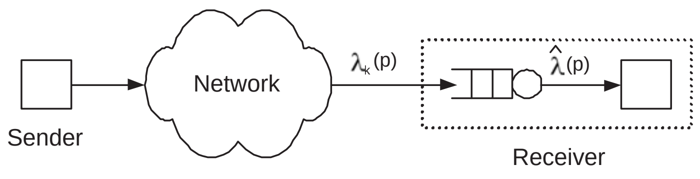


Figure 36: A video streaming model over TCP. $\lambda_k(p)$ stands for the packet arrival rate and $\widehat{\lambda}(p)$ for the playout rate at a receiver.

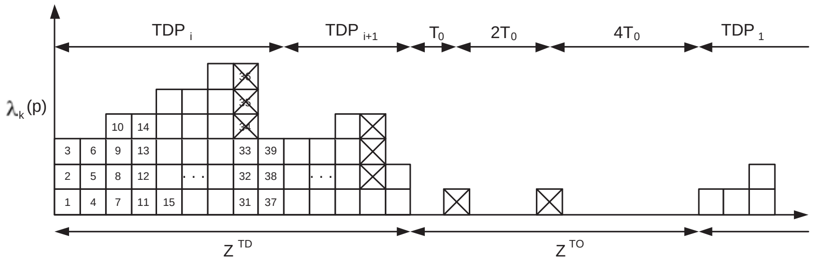
Let $\lambda_k(p)$ be the arrival rate of video packets at round k and $\widehat{\lambda}(p)$ be the video encoding rate[1], where p is the packet loss rate of a flow from a sender to a receiver, and a *round* is defined by a duration between the transmission of packets and the reception of the first acknowledgment (ACK) in a congestion window. It is assumed that a round is equal to the round-trip time (RTT) and independent of the congestion window size.

Figure 37 (a) shows a typical behavior of a TCP flow. We consider the TCP Reno model, since it is one of the most popular implementations in the current Internet [55]. In this model, the steady state throughput is determined by the congestion window size which is adjusted by packet losses. A packet loss can be detected by either triple-duplicate ACKs or timeouts, where we denote the former events by *TD* and the latter by *TO*.

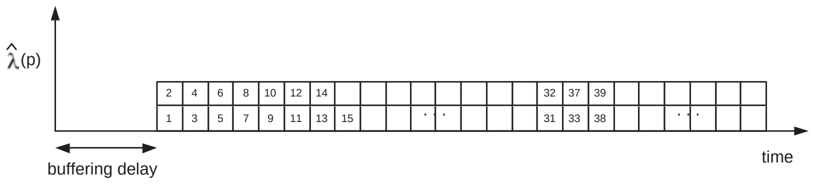
Consider a TD period (TDP) in Figure 37 (a). Each TDP starts immediately after triple-duplicate ACKs and increases the congestion window size by $1/b$ until triple-duplicate ACKs are encountered again. However, when multiple packets are lost and less than three duplicated ACKs are received, a TO period (TOP) begins. In each TOP, a sender stops transmitting data packets for a timeout interval and retransmits non-acknowledged packets. Note that the timeout interval in a TOP increases exponentially until it reaches $64T_0$.

On the other hand, Figure 37 (b) shows the playout characteristic at a receiver, where it is assumed that the video playout rate is two packets worth of data per RTT. We can observe that, if a right size of receiver buffering is employed, a consistent CBR playout can be achieved without any interruption. Since quality degradations are generally caused by frequent buffer underruns that result in playout disruptions or freezings of images, we are interested in providing consistent video quality with interruptions as small as possible.

[1]Note that $\lambda_k(p)$ is a function of time specified by a round k, whereas there is no subscript on $\widehat{\lambda}(p)$, since the data rate fed into a video decoder is assumed to be CBR.



(a) Packet arrival characteristic at a receiver



(b) Playout characteristic at a decoder

Figure 37: An illustration of receiver buffering, where the '×' marks specify lost packets. The packet arrival rate in (a) demonstrates the variability of TCP. On the other hand, (b) shows the consistent playout rate by exploiting right size of buffering delay.

In this research, the performance of a video streaming application is evaluated by the buffer underrun probability and the disruption frequency:

- The *buffer underrun probability* is defined by n/N, where n is the number of buffer underrun events, and N is the number of epochs in a video. An *epoch* is defined by the average of $Z^{TD} + Z^{TO}$, where Z^{TO} is the duration of a TOP, and Z^{TD} is the duration between two TOPs. Note that a Z^{TD} consists of one or more TDPs.

- The *disruption frequency* is defined by n/T [69], where n is the number of buffer underrun events, and T is the duration of a video streaming session. Since T consists of N epochs, a disruption frequency can be expressed by the ratio of the buffer underrun probability to the duration of an epoch.

Since a receiver is either in a TDP or a TOP, the buffer underrun probability at time t is defined by the sum of conditional probabilities, such that

$$P\{q_{min} \leq 0\} = P\{q_{min} \leq 0 | t \in Z^{TD}\}P\{t \in Z^{TD}\} + P\{q_{min} \leq 0 | t \in Z^{TO}\}P\{t \in Z^{TO}\}. \quad (2)$$

Since it is assumed that $\lambda_k(p)$ packets are received, and $\widehat{\lambda}(p)$ packets are drained at round k, the playout buffer size is given by

$$q_k = q_{k-1} + \lambda_k(p) - \widehat{\lambda}(p), \tag{3}$$

where $k = 1, 2, \ldots, X_i$; X_i is the number of round where a TD loss is detected; and q_0 is the playout buffer size at round 0. Notations are summarized in Table 2.

Table 2: Notations

q_0	playout buffer size at round 0
q_k	playout buffer size at round k
q_{min}	minimum buffer size
p	packet loss rate
R	round-trip time
$\lambda_k(p)$	arrival rate of video packets at round k: $\lambda_k(p) = \begin{cases} \frac{W_{i-1}(p)}{2} + \frac{k}{b} - 1 \text{ packets/RTT}, & \text{in TDP } i \\ 0, & \text{otherwise.} \end{cases}$
$\widehat{\lambda}(p)$	video encoding rate
b	number of packets that are acknowledged by an ACK
W_i	congestion window size at the end of TDP i
X_i	number of round where a TD loss is detected
Y_i	number of packets sent in TDP i
α_i	the first packet lost in TDP i
β_i	number of packets sent in the last round
T_0	retransmission timeout
P_u	desired buffer underrun probability
W_m	maximum window size

In a TDP, since the packet arrival rate is greater than zero, and the rate is increased linearly until it encounters triple duplicate ACKs, the playout buffer size at round k is

$$\begin{aligned} q_k &= q_0 + \sum_{n=1}^{k} [\lambda_n(p) - \widehat{\lambda}(p)] \\ &= q_0 + \frac{k^2}{2b} + \frac{[W_{i-1}(p) - 2\widehat{\lambda}(p) - 1]k}{2}. \end{aligned} \tag{4}$$

On the other hand, since all packets are lost in a TOP, and no packet is delivered to a receiver, the playout buffer size at round k in a TOP is[2]

$$\begin{aligned} q_k &= q_{k-1} - \widehat{\lambda}(p) \\ &= q_0 - k\widehat{\lambda}(p). \end{aligned} \tag{5}$$

[2]We assume that a round in Z^{TO} is equal to RTT, although no ACK packet is received during Z^{TO}.

4.2.2 When TCP Throughput Matches Video Encoding Rate

We first investigate the performance when TCP throughput matches video encoding rate. This is the case when the video encoding rate is determined by the access link bandwidth, and the available bandwidth is limited by the access link capacity. For example, many video streaming websites provide multiple copies with identical content, generated at different data rates. A receiver selects an appropriate stream based on the access link capacity.

Since we assume that the video encoding rate is equal to the TCP throughput and does not change over time, the encoding rate is given by [55]

$$\widehat{\lambda}(p) = \frac{1}{\sqrt{\frac{2bp}{3}} + \frac{T_0}{R}\min(1, 3\sqrt{\frac{3bp}{8}})p(1+32p^2)} \text{packets/RTT.}$$

From conditional buffer sizes in (4) and (5), we can derive the buffer requirement under which the probability that the *unconditional* minimum buffer size goes non-positive. Equation (6) states that, given a network condition characterized by the packet loss rate (p) and RTT (R), the playout buffer size that achieves desired buffer underrun probability (P_u) is

$$q_0 \geq \frac{0.16}{pP_u}[1 + \frac{9.4}{b}(\frac{T_0}{R})^2 \min(1, 3\sqrt{\frac{3bp}{8}})p(1+32p^2)]. \qquad (6)$$

The proof of (6) is given in Appendix D.1.

Given playout buffer size, required buffering delay is

$$d_0 = \frac{q_0}{B(p,R)}, \qquad (7)$$

where $B(p,R)$ is the steady state TCP throughput in [55]. Therefore, d_0 corresponds to the time delay for buffered packets to be drained.

The duration of an epoch is also given in [55]. An epoch for a congestion limited TCP flow is given by

$$E[Z^{TD} + Z^{TO}] = \frac{R(\sqrt{\frac{2b}{3p}} + 1)}{\min(1, 3\sqrt{\frac{3bp}{8}})} + T_0\frac{f(p)}{1-p}, \qquad (8)$$

where $f(p) = 1+p+2p^2+4p^3+8p^4+16p^5+32p^6$. Note that a TCP connection is defined by *congestion limited* if the congestion window size is greater than the sender/receiver window size. Otherwise, it is defined by *window limited*. We will investigate the effect of the window size limitation in Section 4.2.4.

4.2.3 Impact of TCP Throughput Under-Provisioning

When a streaming flow is congested inside network, and TCP throughput is decreased less than the video encoding rate, it is difficult to achieve desired video quality. The only solution is to employ a large amount of buffering delay so that a playout buffer not only accommodates TCP rate variability but also prefetches part of a video stream. In this section, we investigate the effect of TCP throughput under-provisioning and derive the buffer size requirement to achieve desired buffer underrun probability.

Equation (9) shows required buffer size to achieve desired buffer underrun probability.

$$q_0 \quad \geq \quad \frac{0.16}{pP_u}[1+\frac{9.4}{b}(\frac{T_0}{R})^2 \min(1,3\sqrt{\frac{3bp}{8}})p(1+32p^2)] + \frac{\sqrt{\frac{2b}{3p}}(\widehat{\lambda}(p) - E\{\lambda_k(p)\})}{P_u \min(1,3\sqrt{\frac{3bp}{8}})}, \quad (9)$$

where we assume the average TCP throughput $E\{\lambda_k(p)\}$ is smaller than the video encoding rate $\widehat{\lambda}(p)$. The proof is given in Appendix D.2.

Compared with (6), (9) shows that required buffer size is increased by the second additional term. This term is proportional to the difference between the video encoding rate and TCP throughput. Therefore, as the difference increases, required buffer size also increases. Note that the duration of an epoch is the same as (8), since it is congestion limited.

4.2.4 Impact of Window Size Limitation

TCP throughput is sometimes limited by the receiver or sender side buffer size. The window size advertisement from a receiver was designed for end-to-end flow control, such that a sender should not transmit more data than can be accommodated by a receiver's capability.

Some video streaming applications transmit packetized video at the data rate at which it was encoded. In this scenario, a sender does not exploit the whole available bandwidth, even the amount of available bandwidth is greater than the video encoding rate. Note that this behavior exhibits the same effect as the sending buffer limitation, since packets are trickled into the network at the data rate of the window size per RTT.

Equation (10) shows required buffer size to achieve desired buffer underrun probability. Observe that it has a simple form compared with congestion limited flows.

$$q_0 \geq \frac{b}{8P_u}(W_m + 1)^2 \tag{10}$$

We present the proof in Appendix D.3.

The duration of an epoch in window limited flows is given in [55], such that

$$E[Z^{TD} + Z^{TO}] = \frac{R(\frac{b}{8}W_m + \frac{1-p}{pW_m} + 2)}{\min(1, \frac{3}{W_m})} + T_0 \frac{f(p)}{1-p}. \tag{11}$$

4.3 Evaluation

In this section, we present simulation results by which playout disruption characteristics are evaluated. The experimental results including simulation scripts are available in the companion website [15].

4.3.1 Experimental Setup

TCP throughput dynamics are generated over a single bottleneck topology. The number of TCP streaming flows is set to 5 in each topology. All access links have sufficient capacity so that any packet drop occurs at the bottleneck link: the access links have 100Mbps capacity and 2ms delay, whereas the bottleneck link has 10Mbps capacity and 40ms delay.

We run `ns-2` [54] simulations over this topology. To model the TCP throughput dynamics, we use the throughput experienced between streaming senders and receivers: the throughput is measured by counting the number of packets delivered to a receiver. All data packets are 1200 bytes long. The queue management algorithm running on intermediate routers is RED, where `min_thresh`, `max_thresh`, the queue size of a router are 0.25, 1.25, and 2.5 times of the bandwidth delay product respectively, as suggested in [4].

To construct dynamic network characteristics, competing traffic (or cross traffic) is generated by triggering persistent FTP flows 10 seconds prior to TCP streaming sessions. The number of cross traffic flows is varying to investigate the effect of the packet loss rate on the performance of TCP streaming. Unless otherwise specified, following sets of configurations are examined, each of which generates 10 traces using random seeds.

- *Configuration 1*: the number of competing FTP flows is assumed to be 5 that leads to 122.5ms RTT, 0.8% packet loss rate, and 1Mbps throughput on average. The duration of simulation time is set to 600 seconds.
- *Configuration 2*: the number of competing flows is set to 10. Measured network characteristics are 130.6ms RTT, 1.43% packet loss rate, and 666.3kbps throughput. We employ 600 seconds simulation time.
- *Configuration 3*: the number of competing flows is 15. Measurement results are 138.6ms RTT, 2.05% packet loss rate, and 499.4kbps throughput. Simulation time is also 600 seconds.

In each configuration, TCP throughput is measured by counting the number of delivered packets. To estimate the packet loss rate of a congestion limited flow, we employ the TCP throughput equation in [55]. The equation provides an analytic relationship between the packet loss rate, RTT, and TCP throughput. However, as the relationship is too complicated to yield a closed form of a packet loss rate as a function of throughput and RTT, we construct an iterative algorithm based on the bisection method [58]. Since the TCP throughput equation is continuous and an estimated throughput must lie in the packet loss rate of [0, 1], the existence of a root is guaranteed by the intermediate value theorem. Also the estimated packet loss rate is unique, since the estimated throughput is monotonically decreasing as the packet loss rate increases.

The performance of TCP streaming experiments is evaluated by the buffer underrun probability and the disruption frequency, defined in Section 4.2.1. Note that, since the simulation time in each simulation trace is 600 seconds, the number of epochs is the simulation time divided by the duration of an epoch.

4.3.2 Experiment 1: Matching TCP Throughput

In the first experiment, we assume that the average TCP throughput matches video encoding rate. We investigate 50 TCP streaming flows, since a configuration contains 10 traces, each of which contains 5 TCP streaming flows.

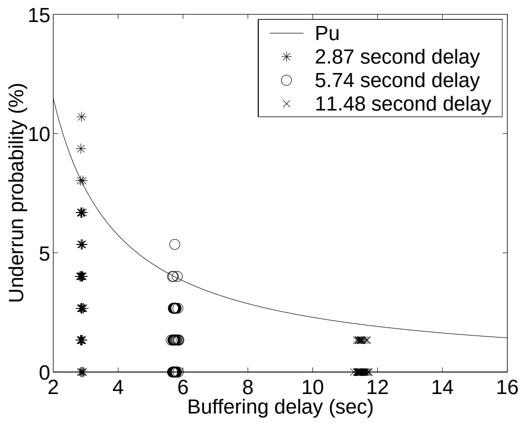
Figure 38 (a) shows the buffer underrun characteristics of configuration 1. The solid line specifies the minimum buffering delay requirements to achieve desired buffer underrun probability. Each marker corresponds to a measurement result that exhibits a specific number of buffer underrun events in a TCP streaming flow. Using (7), when RTT and the packet loss rate are 122.5ms and 0.8%, buffering delays targeting desired buffer underrun probabilities of 8%, 4%, and 2% are 2.87, 5.74, and 11.48 seconds respectively. Experimental results show that most of measured buffer underrun probabilities are distributed below the solid line, and it is observed that 96% of all flows exhibit equal or less than desired buffer underrun probability when buffering delay is 2.87 seconds (i.e., buffer underrun probabilities of 48 flows are at most 8%). The accuracy is improved as buffering delay is increased: 98% flows achieve 4% buffer underrun probability for the 5.74 second delay, and all flows achieve less than 2% buffer underrun probability for the 11.48 second delay.

Note that the buffering delay characteristic is a non-linear curve. For example, when buffering delay is increased from 4 seconds to 10 seconds, desired buffer underrun probability is reduced by 3.5%. However, when buffering delay is increased from 10 seconds to 16 seconds, the probability is reduced by only 0.8%. Therefore, a system designer can find a point of marginal return using the non-linear characteristics.

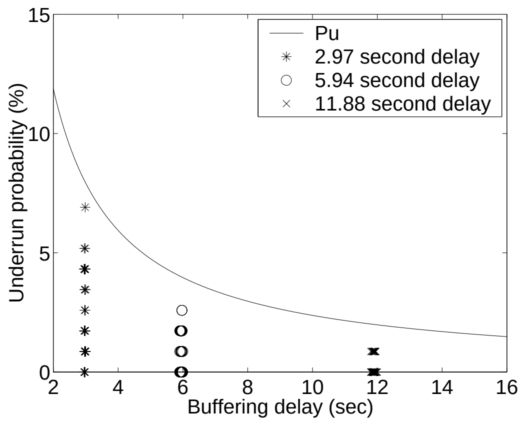
Figure 38 (b) shows the buffer underrun probability of configuration 2. Required buffering delays targeting P_u = 8%, 4%, and 2% are 2.97, 5.94, and 11.88 seconds respectively. Compared with Figure 38 (a), the accuracy of analytic performance is improved: all flows achieve less buffer underrun probabilities than desired buffer underrun probabilities.

Figure 38 (c) shows the buffer underrun probability of configuration 3. Required buffering delays for P_u = 8%, 4%, and 2% are 3.42, 6.84, and 13.68 seconds respectively. Experimental results demonstrate the same characteristics as Figure 38 (b), such that all flows achieve desired buffer underrun probability, and measured buffer underrun probabilities are further reduced.

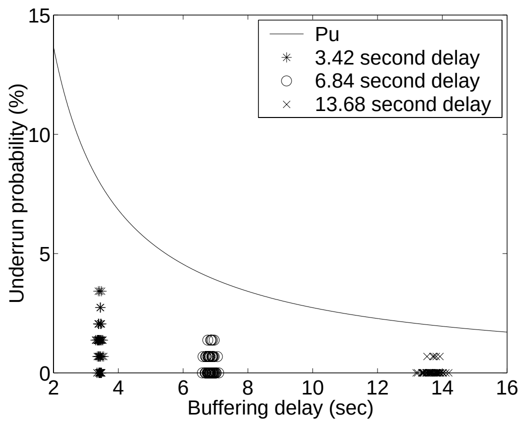
Figure 39 shows disruption frequency characteristics for configuration 1, 2, and 3. Since the disruption frequency is expressed by the ratio of the buffer underrun probability to the duration of an epoch, it exhibits the same characteristics as Figure 38, scaled by the



(a) Configuration 1

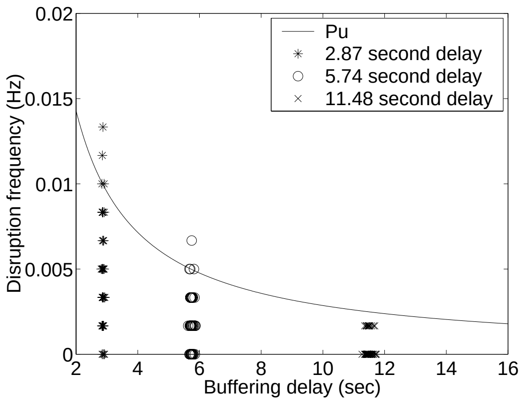


(b) Configuration 2

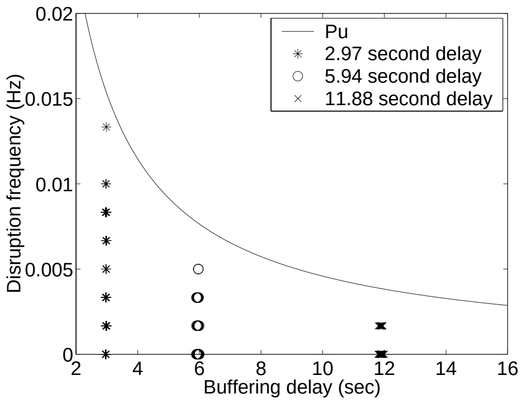


(c) Configuration 3

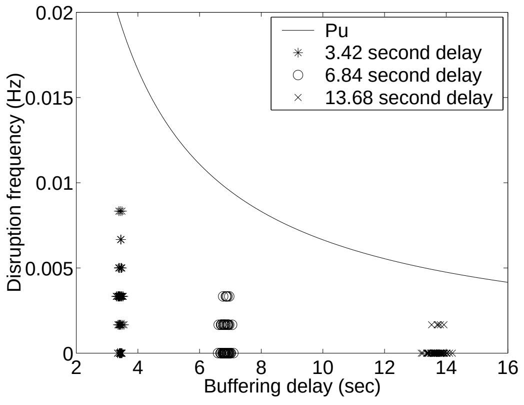
Figure 38: Measured buffer underrun probability when TCP throughput matches video encoding rate. Solid line specifies desired buffer underrun probability. Each marker specifies a measured buffer underrun probability of a TCP flow.



(a) Configuration 1



(b) Configuration 2



(c) Configuration 3

Figure 39: Measured disruption frequency when TCP throughput matches video encoding rate. Note that the disruption frequency shows the same characteristic as the buffer underrun probability.

epoch length. Note that the disruption frequency is further decreased when packet loss rate is small, since small packet loss rate yields a long epoch. For example, $P_u = 8\%$ in configuration 1 results in 0.01Hz disruption frequency. On the other hand, $P_u = 8\%$ in configuration 2 and 3 leads to 0.015Hz and 0.019Hz disruption frequencies.

4.3.3 Experiment 2: TCP Throughput Under-Provisioning

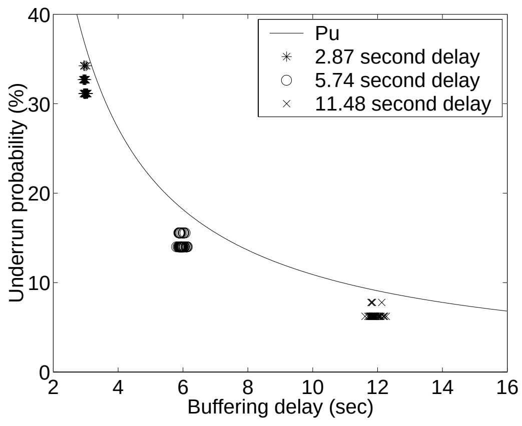
In the second experiment, we assume that TCP throughput is smaller than the video encoding rate by 10%: the encoding rates of configuration 1, 2, and 3 are assumed to be 1.1Mbps, 732.9kbps, 549.3kbps, respectively. However, it is assumed that buffering delays are the same as in Section 4.3.2.

Figure 40 (a) shows the buffer underrun probability of configuration 1. From (9), the number of buffer underrun events is increased significantly, since TCP throughput is not sufficient to accommodate the encoded video. Compared with Figure 38 (a), desired buffer underrun probabilities are increased by 4.75 times. In Figure 40 (b), desired buffer underrun probabilities are 3.21 times greater than Figure 38 (b). Figure 40 (c) demonstrates similar characteristics: desired buffer underruns are increased by 2.79 times on average.

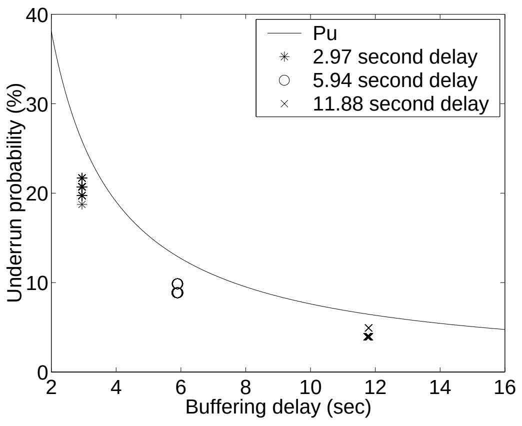
Note that slight under-provisioning of available bandwidth incurs significant increase of buffer underrun events. Conversely, slight increase of the video encoding rate is critical to perceived visual quality: it is expected from experimental results that, when streaming a CBR video, 10% under-provisioning of available bandwidth will incur more than twice of buffer underrun events, and therefore perceived quality will be degraded significantly. Therefore, end users should try to make a video streaming application operate on a well provisioned environment for the maximum video quality.

4.3.4 Experiment 3: Window Size Limitation

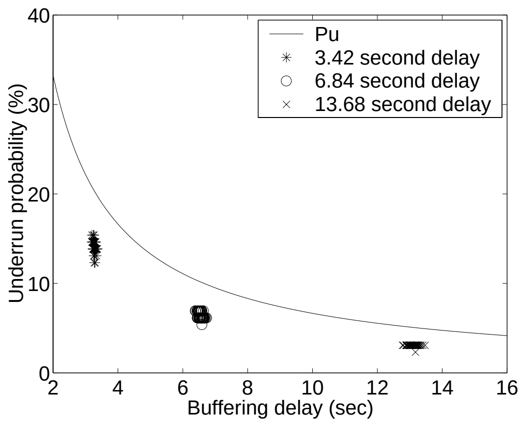
In this experiment, we investigate the effect of the window size limitation on playout disruptions. We assume that TCP streaming sessions are generated from persistent and buffer-limited TCP flows with the maximum window size of $W_m = 12$. For fair comparison with the results of Section 4.3.2 and 4.3.3, we employ different experiment settings from Section 4.3.1, since the characteristic change of TCP streaming flows leads to the change of



(a) Configuration 1



(b) Configuration 2



(c) Configuration 3

Figure 40: Measured buffer underrun probability when TCP throughput is less than video encoding rate by 10%. Since TCP throughput is not sufficient to accommodate the encoded video, the buffer underrun probability is increased significantly.

measured network characteristics. In this section, dynamic network characteristics are generated by a link error model, such that the bottleneck link uniformly randomly discards arriving packets, while competing traffic is disabled. The loss rate is set to 0.8%, measured RTT is 89.7ms, and measured throughput is 1Mbps. Simulation time is set to 600 seconds, and the duration of epoch is given by (11).

Figure 41 shows that measured buffer underrun probabilities are decreased significantly. Compared with Figure 38 (a), measured buffer underrun probabilities are reduced by at least four times. This is not surprising, since TCP flows with the window size limitation exhibit highly smooth throughput, and therefore the TCP throughput variability can be easily accommodated.

To assess the throughput variability of a congestion limited TCP flow and a window limited TCP flow, we compare the coefficient of variation (CoV), which is defined by the ratio of standard deviation to average of TCP throughput. Our measurements show that measured CoV of TCP flows in this section (window limited flows) is 0.31. On the other hand, measured CoV of the flows in Section 4.3.2 (congestion limited flows) is 0.44 which accounts for 1.4 times larger throughput variability.

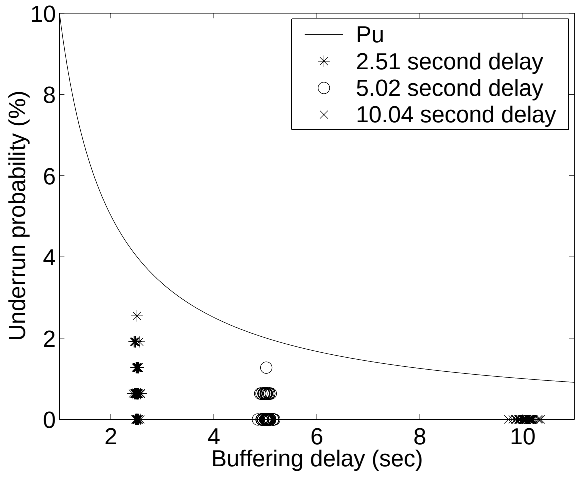


Figure 41: Measured buffer underrun probability when TCP throughput is limited by the maximum window size. Experiments are carried out over a link error model of the `NS-2` simulator.

4.3.5 Summary of Results

In our simulation study, we investigate buffer underrun characteristics to achieve desired buffer underrun probability. Experimental results show that, when TCP throughput matches video encoding rate, measured buffer underrun probabilities of more than 96% of streaming flows are smaller than desired buffer underrun probabilities. In the case of TCP throughput under-provisioning, measured buffer underrun probabilities are increased significantly: 10% of bandwidth under-provisioning results in at least 2.8 times more buffer underruns. When TCP throughput is limited by the maximum window size, it is observed that measured buffer underrun probabilities are decreased, since TCP throughput variability is reduced. Still, measured buffer underrun probabilities are smaller than desired buffer underrun probabilities in both scenarios.

4.4 Internet Experiments

In this section, we present the Internet experiment results to validate our analysis and simulation study in Sections 4.2 and 4.3.

4.4.1 Experiment Methodology

We first describe the Internet experiment methodology to evaluate the performance of video streaming over TCP. We investigate playback disruption characteristics using *Windows Media Player* [52], since it is one of the most popular video streaming players [74]. Windows Media Player employs the real-time streaming protocol (RTSP) and the multimedia server (MMS) protocol to communicate with a video streaming server. The application level information, such as the data reception rate, the video encoding rate, or streaming protocol, are measured using the *MediaTracker* tool [48]. Packet level information, such as IP address or port number, can be measured by the *Ethereal* network protocol analyzer [22]. We examine following video streams:

1. *Windows XP launch webcast*[3] which is encoded at 305.2kbps and delivered over RTSP/TCP.

[3]`http://www.microsoft.com/windowsxp/wmx/launch300k.asx`

2. *Comdex keynote speech*[4] in which the clip is encoded at 290.9kbps and delivered over RTSP/TCP.

3. *Public lecture series – "Israel: peace and war"*[5] which is encoded at 297.9kbps and delivered over MMS/TCP.

In all experiments, buffering delay is set to 5 seconds.

4.4.2 Experimental Results

Table 3 summarizes the Internet experiment results of the three video clips. Each row in the table corresponds to a TCP streaming connection, measured on a day in January 2004. The second column indicates a date when a measurement was carried out. The third column shows the duration of a TCP streaming session. The fourth column is the data rate at which a video is encoded. The next three columns describe network characteristics, such as network throughput, RTT, and the packet loss rate. The eighth column specifies desired buffer underrun probabilities determined by (6), (9), or (10). Finally, the last column reports measured buffer underrun probabilities defined in Section 4.3.1.

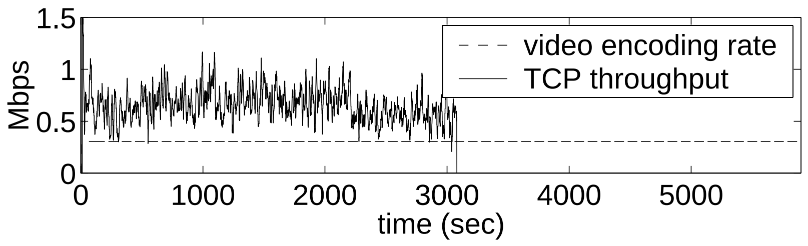
Table 3: Internet experiment results

stream	date	duration (sec)	encoding rate (kbps)	throughput (kbps)	RTT (ms)
1	1/4/2004	5898	305.2	655.8	80.6
	1/15/2004	2199	305.2	276.1	124.1
2	1/4/2004	3994	290.9	1517.4	63.3
	1/10/2004	1201	290.9	289.2	100.3
3	1/4/2004	4697	297.9	303.1	24.4
	1/16/2004	4702	297.9	303.4	24.5

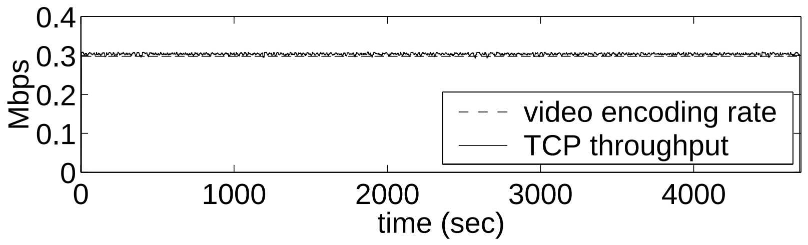
stream	packet loss rate (%)	P_u (%)	measured underrun probability (%)
1	3.02	–	0.0
	5.18	8.1	4.4
2	1.22	–	0.0
	6.20	8.3	2.6
3	–	–	0.0
	–	–	0.0

[4] `http://metahost.cwusa.tv/microsoft/20031116/comdex_keynote_300.asx`
[5] `http://www.princeton.edu/WebMedia/lectures/20031110ozVN300K.asx`

We first investigate the behavior of the RTSP and MMS protocols over TCP. Figure 42 shows the video playout rate and network throughput measured on January 4, 2004. Since the date was a weekend, we expected large amount of network throughput. Figure 42 (a) shows that RTSP exploits available bandwidth aggressively: a receiver receives video packets at a faster rate than the video encoding rate whenever possible, stores them in disk cache, and displays them later. Measurement result shows that the receiver downloaded and displayed the video clip until 3080 seconds. After finishing downloading, it played out the cached video. On the other hand, Figure 42 (b) shows the measurement result of the MMS protocol over TCP. Although there was plenty of available bandwidth, MMS exploited it conservatively: packets were delivered at almost the same rate as the video encoding rate.



(a) RTSP over TCP



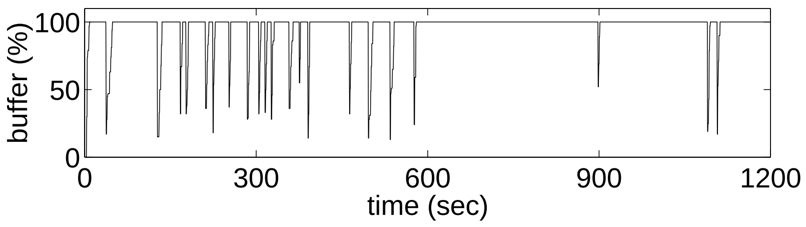
(b) MMS over TCP

Figure 42: Comparison of the RTSP and MMS protocols over TCP. RTSP exploits available bandwidth aggressively, whereas MMS exploits it conservatively.

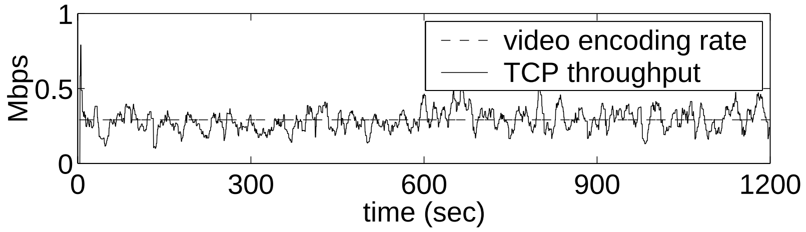
The third and fifth rows in Table 3 show the playout disruption characteristics over congestion limited connections. Measurement results show that the buffering delay requirements in Section 4.2 hold well in the Internet experiments: measured buffer underrun probabilities are bounded by desired buffer underrun probabilities.

Note that the video stream 3 did not show any buffer underrun. We reckon this is due to the Internet 2 connection between the authors' institute and the Princeton university. Also note that we cannot estimate the packet loss rate of it using the estimation algorithm in Section 4.3.1, since it is not congestion limited.

Figure 43 shows measured TCP throughput and buffer occupancy on January 10. Each glitch in Figure 43 (a) corresponds to a buffer underrun event. Figure 43 (b) shows that TCP throughput was consistently fluctuating around the video encoding rate. However, it exhibits the existence of a large time-scale variability: the average TCP throughput was 268.4kbps in [0, 600] seconds, as opposed to 309.9kbps in [600, 1200] seconds. Since the TCP throughput during the [0, 600] second interval was 8% smaller than the video encoding rate, it is observed that measured buffer underrun probability was 4.1%. On the other hand, only three buffer underrun events were observed during [600, 1200] seconds.



(a) Buffer occupancy



(b) Video encoding rate and TCP throughput

Figure 43: Internet experiment results on January 10. Measured buffer underrun probability in (a) is strictly smaller than desired buffer underrun probability.

4.4.2.1 Summary of Results

We investigate the buffer underrun characteristics of video streaming over TCP through the Internet experiments. Experimental results show that, when available network bandwidth is significantly greater than video encoding rate, no buffer underrun event is observed. On the other hand, when network throughput is smaller than video encoding rate, measured buffer underrun probability is greater than zero. Still, the probability is bounded by the desired buffer underrun probability in all cases. These results validate our analysis in Sections 4.2 and 4.3.

4.5 Related Work

Challenges and efforts in streaming video over TCP have been addressed in the literature. Authors in [79] proposed a TCP variant in which a receiver measures data reception rate, and a sender adjusts window size according to the data rate reported by the receiver. However, it requires kernel modification which is inconvenient and may cause bugs in an operating system. Authors in [29] proposed another TCP variant in which a receiver controls throughput by delaying ACK packets. However, this scheme requires both kernel modification and network support (i.e., congestion notification signal, such as ECN). The author in [6] proposed a stochastic model of TCP and derived a condition under which no playback disruption is observed. However, it requires to solve complicated differential equations and to estimate stochastic model parameters.

To overcome the burstiness of TCP, TCP friendly rate control protocols have been proposed. The authors in [25] developed a TFRC protocol in which a flow estimates throughput by measuring average RTT and average packet loss rate. Since the flow adapts the data rate based on average throughput, it is slowly responsive to packet losses, and therefore throughput variability can be reduced. The authors in [3] proposed binomial congestion control algorithms which are TCP friendly and exhibit smooth throughput variability. Video streaming experiments over TCP and a TCP friendly transport protocol were conducted in [71]. The authors demonstrated the superiority of the TCP friendly protocol when combined with an error-resilient scalable codec. However, they did not investigate the effect of receiver buffering on video quality.

Receiver buffering is a well-known technique to accommodate data rate variability. The authors in [7] established a theoretic foundation of performance bounds. They derived the backlog bound (or the maximum buffer size) of a guaranteed service in general communication networks. However, it is difficult to use for video streaming over TCP, since they considered a highly abstract model. The authors in [65] developed a work-ahead smoothing algorithm for optimal rate smoothing. When used with guaranteed service or RCBR service [27], it was demonstrated that the algorithm increases network utilization significantly. However, the algorithm requires the bandwidth evolution information in advance which is not available in best effort networks.

4.6 Conclusion

In this chapter, we consider video streaming over TCP. While the use of TCP provides reliable video stream delivery, the bursty nature of TCP requires buffering at a receiver for smooth video playout. Since it is desirable to determine the right size of playout buffer in memory-constrained devices, we quantify buffering requirements to achieve desired buffer underrun probability by analytically modeling a video streaming system. Our model takes account of three scenarios: 1) when TCP throughput matches video encoding rate, 2) when TCP throughput is smaller than the encoding rate, and 3) when throughput is limited by the maximum window size. Experimental results of both simulations and Internet measurements validate our model.

We believe that the results presented in this chapter can be employed by a number of applications. For example, buffering delays can be adjusted under dynamic network conditions. By dynamically equalizing buffering delays, a receiver will reduce the number of buffer underruns. Also it will avoid unnecessarily long buffering delay, when network condition is improved.

CHAPTER 5

REALTIME ESTIMATION OF PLAYOUT BUFFER REQUIREMENTS IN VIDEO STREAMING OVER TCP

5.1 Motivation

A video streaming application has to employ a transport layer protocol to transmit packetized video data. Typically the protocol provides a mechanism for flow/congestion control and plays a role in avoiding the congestion collapse of the Internet [24]. Since TCP is the dominant protocol in the Internet, it is reasonable to employ TCP for video streaming. Especially, when video streaming servers are placed behind firewalls that permit only pre-specified port numbers, video streaming over TCP is the only choice to get around the firewalls using well-known port numbers.

However, TCP has been regarded inappropriate for video streaming. The main arguments are: 1) the sawtooth behavior of additive increase and multiplicative decrease (AIMD) of TCP incurs significant data rate variability and degrades perceptual video quality, and 2) the use of retransmission timeouts may introduce unacceptable end-to-end delay, and the retransmitted data may be delivered too late for display.

The rate variability of TCP caused by the burstiness and retransmission timeouts can be mitigated through receiver-side buffering. The buffer size has to be large enough to insure that no video data is lost. However, in memory-constrained devices, it is important to determine the right size of playout buffer to provide a prescribed video quality.

In this research, we consider how to realize the provisioning of playout buffer requirements in realtime. Our starting point is the analytic model of a video streaming system over TCP in Chapter 4. Using the model, the buffer size requirements can be quantified as follows: when TCP throughput matches video encoding rate, desired playout buffer size is

$$q_0 \geq \frac{0.16}{pP_u}[1 + \frac{9.4}{b}(\frac{T_0}{R})^2 \min(1, 3\sqrt{\frac{3bp}{8}})p(1 + 32p^2)], \tag{12}$$

where q_0 stands for desired playout buffer size, p for the packet loss rate, R for the round-trip time (RTT), T_0 for the retransmission timeout, and P_u for desired buffer underrun probability. The buffer underrun probability is defined by n/N, where n is the number of buffer underrun events, and N is the number of epochs in a video. An epoch is defined by the average duration of a triple duplicate ACK period and a timeout period in Chapter 4.

Note that we assume the operating point of TCP throughput matches the video encoding rate. This is the case when the video encoding rate is determined by the access link bandwidth, and the available bandwidth is limited by the access link capacity.

Based on the relationship in (12), we propose realtime estimation algorithms of playout buffer requirements in video streaming over TCP. Since the Internet does not provide any information about network conditions in advance, we first develop a realtime estimation algorithm of receiver buffer size by estimating packet loss rate and RTT. We further develop realtime throughput adaptation algorithms that dynamically adjust data transmission rate to prevent playback disruptions proactively.

This research is organized as follows. In Section 5.2, we present an estimation algorithm of the playout buffer size without network characteristics information in advance. Section 5.3 present throughput adaptation algorithms to adjust data transmission rate in realtime. Section 5.4 reports on experimental results of video streaming over TCP that we have conducted. The research is concluded in Section 5.5.

5.2 Realtime Estimation of Playout Buffer Size

In this section, we present a realtime estimation algorithm to determine the playout buffer size based on the video streaming model in Chapter 4. This task is challenging, since there is no guideline for the provisioning of buffering size in realtime, even though most video streaming players allow a user to change buffering delay. Most conventional approaches are based on simple trial-and-error (e.g., [52]): to increase the amount of buffering delay until no playback disruption is observed. Although this approach is simple and eventually converges to a desired value, it is inefficient and takes long time to find an optimum.

To realize realtime estimation, we develop an algorithm based on the playout buffer requirements in (12). Our approach is the passive estimation: to keep buffering received packetized video during an estimation period while at the same time required buffer size is estimated.

Figure 44 shows the realtime buffering delay estimation algorithm, where τ stands for the estimation period, t for current time, $\widehat{p}$ for estimated packet loss rate, $\widehat{R}$ for estimated RTT, $\widehat{B}$ for estimated throughput, and $\widehat{q}(\widehat{p}, \widehat{R})$ for estimated buffer size defined by

$$\widehat{q}(\widehat{p}, \widehat{R}) = \frac{0.16}{\widehat{p}P_u}[1 + \frac{9.4}{b}(\frac{\widehat{T_0}}{\widehat{R}})^2 \min(1, 3\sqrt{\frac{3b\widehat{p}}{8}})\widehat{p}(1 + 32\widehat{p}^2)].$$

```
1: procedure ComputeBufferingDelay
2:     gettimeofday(&$t_0$)
3:     $t = t_0$
4:     while $t - t_0 < \tau$ or $t - t_0 < \widehat{d_0}$ do
5:         gettimeofday(&$t$)
6:         $\widehat{p}$ = EstimatePacketLossRate()
7:         $\widehat{R}$ = EstimateRTT()
8:         $\widehat{d_0} = \widehat{q}(\widehat{p}, \widehat{R})/\widehat{B}$
9:     enddo
10: endprocedure
```

Figure 44: Realtime estimation algorithm of playout buffering delay. A receiver keeps buffering during the estimation period.

The algorithm estimates the buffering delay, d_0, during the estimation period, τ. If the estimated buffering delay is smaller than τ, it ends immediately (line 4). Otherwise, a receiver keeps buffering received packets and estimating the buffering delay until it reaches the estimated delay. Since the buffer size is determined by $\widehat{p}$ and $\widehat{R}$, we need to estimate the packet loss rate and RTT.

The packet loss rate estimation algorithm was proposed in the context of the TFRC protocol in [25], where packet loss rate is estimated by the average loss interval. A loss interval is defined by the number of packets between two loss events, where a loss event is defined by an event of packet loss within a single RTT. TCP senders and receivers measure the loss interval by examining sequence numbers and ACK numbers in the TCP header. Specifically, a sender can detect a loss event if triple-duplicate ACKs are received or timeout occurs in an RTT. The average loss interval is defined by

$$\widehat{s} = \frac{\sum_{i=1}^{n} w_i s_i}{\sum_{i=1}^{n} w_i},$$

where s_i's are measured loss intervals, $n = 8$, $w_1 = w_2 = w_3 = w_4 = 1$, $w_5 = 0.8$, $w_6 = 0.6$, $w_7 = 0.4$, and $w_8 = 0.2$. The estimated packet loss rate is determine by the inverse of the average loss interval, i.e., $\widehat{p} = 1/\widehat{s}$.

The estimated RTT can be measured by the algorithm in [72]. The algorithm estimates RTT and retransmission timeouts using a moving average (MA) estimator, such that

$$\begin{aligned} Err &\leftarrow M - \widehat{R} \\ \widehat{R} &\leftarrow \widehat{R} + gErr \\ D &\leftarrow D + h(|Err| - D) \\ \widehat{T_0} &\leftarrow \widehat{R} + 4D, \end{aligned}$$

where M stands for sampled RTT, $\widehat{R}$ for smoothed RTT, D for mean deviation, $\widehat{T_0}$ for estimated retransmission timeout, $g = 0.125$, and $h = 0.25$. Because of its simplicity and good performance, this algorithm was employed by the TCP protocol specification in [57].

5.2.1 Performance Evaluation

We evaluate the accuracy of the realtime estimation algorithm by `ns-2` [54] simulations. We generate throughput variability resulting from transport protocols over the single bottleneck topology: senders are arranged on one end of a link, and receivers on the other end. All links except the bottleneck link have sufficient capacity so that any packet drop occurs at the bottleneck link. The bottleneck link capacity is set to 10Mbps with 20ms link delay, whereas access links have 100Mbps capacity with 2ms link delay. Background traffic is generated by superposing 100 ON/OFF UDP sources of pseudo nodes that have the Pareto distribution [75]. We generate 20 different traces by changing the duration of ON/OFF period of the UDP traffic. Since each trace contains three TCP streaming flows, we examine 60 flows as a whole. The packet size is set to 1200 bytes in all flows, and the duration of simulation time is set to 180 seconds. TCP throughput is measured by counting the number of packets delivered to TCP receivers.

In Figure 45, we investigate the impact of the estimation period on the accuracy by varying τ into multiples of smoothed RTT (i.e., τ = 30, 40, 50, and $60\widehat{R}$). The error bar specifies the 95% confidence interval. We observe that buffering delay is significantly over-estimated when the estimation period is small ($\leq 40\widehat{R}$). This is because the exponential increase in TCP slow start incurs significant packet losses that lead to throughput under-estimation. However, for large τ ($\geq 60\widehat{R}$), it is observed that more than 80% of estimated buffering delays lie within the desired 5.5 ± 1.1 second range.

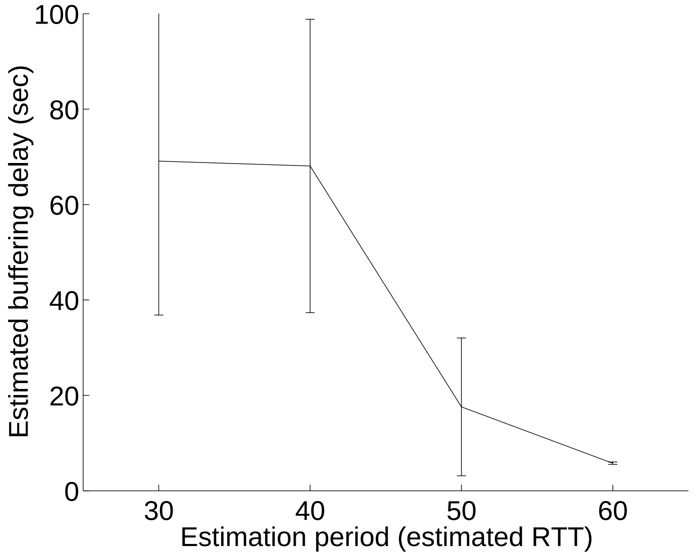


Figure 45: The impact of the estimation period on the accuracy of estimated buffering delay. A large estimation period ($\geq 60\widehat{R}$) yields high accuracy.

It should be noted that a similar observation was reported in [28]. The authors exploited the TCP throughput equation in [55] and estimated packet loss rate and RTT to speedup discrete event simulation. However, they employed tens of triple-duplicate ACK periods for an estimation period because of the high accuracy requirement.

5.3 Realtime Throughput Adaptation

In Section 5.2, it is assumed that an estimated buffering delay does not change in a session, and therefore the estimated packet loss rate and the estimated RTT are assumed to be static. However, since the Internet exhibits dynamic behavior of multiple time-scale variability, we need a mechanism that dynamically adjust data transmission rate to proactively protect receiver buffer from buffer underruns.

In this section, we present realtime throughput adaptation algorithms. Realtime throughput adaptation is especially important when used with rate adaptive video codecs that can dynamically adjust data transmission rate based on network conditions [47], [59], [71]. The basic idea of realtime throughput adaptation is that a sender transmits all data packets when network throughput is greater than video encoding rate. However, when the throughput is decreased, the sender proactively reduces the data transmission rate by discarding part of data packets randomly. This discarded information can be reconstructed in the receiver using an error-resilience module.

5.3.1 Equation-Based Adaptation

We first consider equation-based rate adaptation. In equation-based adaptation, TCP throughput is determined indirectly by estimating RTT and packet loss rate. Data transmission rate is adjusted based on the TCP throughput equation in [55], such that

$$\widehat{B}(\widehat{p},\widehat{R}) = \frac{1}{\widehat{R}\sqrt{\frac{2b\widehat{p}}{3}} + \widehat{T_0}\min(1, 3\sqrt{\frac{3b\widehat{p}}{8}})\widehat{p}(1+32\widehat{p}^2)},$$

where $\widehat{p}$ and $\widehat{R}$ are estimated packet loss rate and smoothed RTT in Section 5.2.

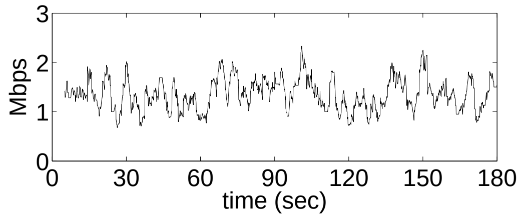
To evaluate the performance of equation-based adaptation, Figure 46 shows the data transmission rate of three TCP flows in Section 5.2, where buffering delay is set to 5.5 seconds. Unfortunately, experimental results show that the data transmission rate exhibits significant variability.

We carry out the sensitivity analysis to investigate the performance of the equation-based adaptation algorithm. Since $\widehat{B}$ can be approximated by $\widehat{B} = \frac{\sqrt{3/2b}}{R\sqrt{p}}$ for small p, we define a sensitivity to the RTT change (S_R) and a sensitivity to the packet loss rate change (S_p) by

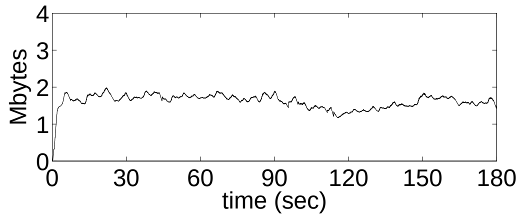
$$S_R = \frac{\Delta\widehat{B}/\widehat{B}}{\Delta R/R}, \tag{13}$$

$$S_p = \frac{\Delta\widehat{B}/\widehat{B}}{\Delta p/p}, \tag{14}$$

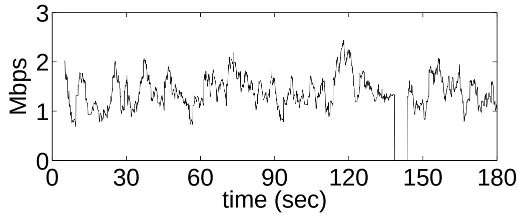
where ΔR stands for the RTT change, Δp for the packet loss rate change, and $\Delta\widehat{B}$ for the



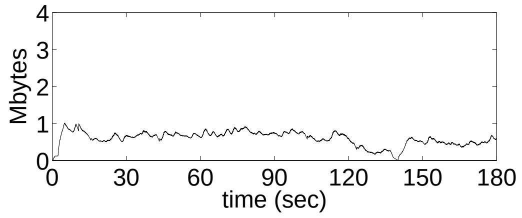
(a) Data transmission rate of TCP 1



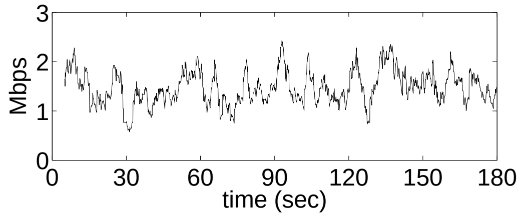
(b) Buffer occupancy of TCP 1



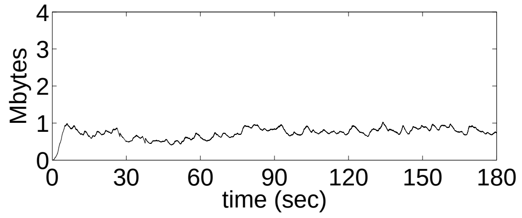
(c) Data transmission rate of TCP 2



(d) Buffer occupancy of TCP 2



(e) Data transmission rate of TCP 3



(f) Buffer occupancy of TCP 3

Figure 46: Performance of the equation-based adaptation algorithm. Data transmission rate exhibits significant variability.

data transmission rate change. Therefore, we can model the data rate variation as

$$\begin{aligned}\widehat{B} + \Delta\widehat{B} &= \frac{\sqrt{3/2b}}{(R+\Delta R)\sqrt{p+\Delta p}} \\ &\approx \frac{\sqrt{3/2b}}{R\sqrt{p}}(1 - \frac{\Delta R}{R} - \frac{\Delta p}{2p}). \end{aligned} \qquad (15)$$

From (13), (14), and (15), it follows that

$$S_R \approx -1, \qquad (16)$$

$$S_p \approx -0.5. \qquad (17)$$

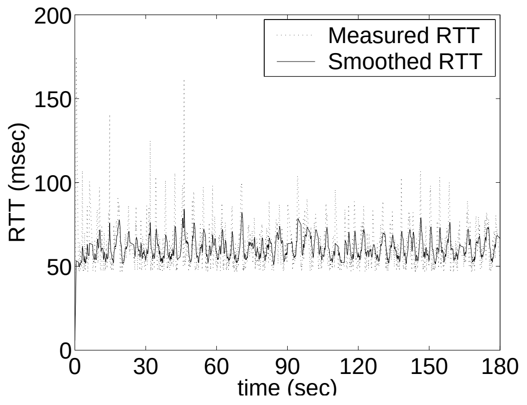
Equations (16) and (17) imply that the amount of RTT change leads to the same amount of the data transmission rate change, and that the packet loss rate change to half of the data transmission rate change. Therefore, even small inaccuracies as much as a few milliseconds of RTT or a few tenth of a percent of packet loss rate are critical to the data transmission rate estimation. Hence, the RTT estimation algorithm and the packet loss rate estimation algorithm should provide high accuracy. Also the estimated RTT and packet loss rate should be highly smooth and do not change over short periods of time, since the variability in estimations leads to data rate variability.

However, these requirements are generally not feasible, since each packet likely experiences different network characteristics. Figure 47 shows the performance of RTT and packet loss rate estimation algorithms in TCP flow 1. Figure 47 (a) shows measured RTT and smoothed RTT. Although the RTT estimation algorithm significantly reduces the variability of measured RTT, smoothed RTT still exhibits variability ranging from 51ms to 84ms. Figure 47 (b) shows measured and estimated packet loss rate. This result demonstrates that estimated packet loss rate exhibits variability ranging from 0.6% to 4.3%.

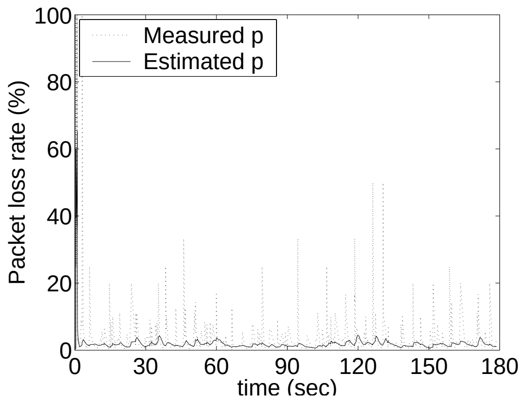
From this observation, we can find that video streaming based on the equation-based adaptation algorithm hardly provides good video quality, since the smoothed RTT and the estimated packet loss rate exhibit significant variability.

5.3.2 Direct Estimation Algorithm

Since the dependency on RTT and packet loss rate estimation leads to significant data rate variability in equation-based adaptation, we develop a straightforward approach to adjust



(a) RTT estimation



(b) Loss rate estimation

Figure 47: Performance of RTT and packet loss rate estimation algorithms. Even though the algorithms yield smoothed RTT and packet loss rate, they still exhibit variability.

data transmission rate. In this section, we directly estimate TCP throughput without considering RTT and packet loss rate estimations.

In direct estimation, we define the *measurement period* T by a time period over which the ith throughput sample $\widehat{B}_i$ is averaged with the amount of received data, i.e., $\widehat{B}_i = \frac{1}{T}\sum_{k \in I_i} \lambda_k(p)$, where $\lambda_k(p)$ is the packet arrival rate in round k, I_i is the ith measurement event, and $|I_i| = T$. Based on throughput samples, TCP throughput is determined by a linear estimator, such that

$$\widehat{B} = \frac{\sum_{i=1}^{n} w_i \widehat{B}_i}{\sum_{i=1}^{n} w_i},$$

where the coefficients, n and w_i, are the same as those of Section 5.2. Since $\widehat{B}_i$ is sampled at the end of a measurement period, the estimated throughput $\widehat{B}$ is updated less frequently than equation-based adaptation when a measurement period is large. Therefore, the accuracy of direct estimation depends on whether the measurement period is large enough to exhibit small sampling error. Note that the measurement period should not be larger than the buffering delay. Otherwise the data transmission rate between the buffering delay and the first measurement period cannot be determined.

Figure 48 shows the performance of the direct estimation algorithm. The algorithm is applied to TCP flow 1 in Section 5.2. The amount of the buffering delay is set to 5.5 seconds. This result shows that the direct estimation algorithm reduces the data transmission rate variability significantly, and the variability keeps decreasing as the measurement period increases. This is because larger measurement period exploits more throughput samples which lead to reduced statistical sampling error.

We evaluate the variability by the coefficient of variation (CoV) performance which is defined by the ratio of standard deviation to average. Note that a smaller CoV implies a smoother flow. In Figure 48, CoVs of the data transmission rate for $T = 1, 2, 3$, and 4 seconds are 0.18, 0.13, 0.12, and 0.11 respectively, whereas the average CoV of three data transmission rates of equation-based adaptation in Figure 46 is 0.25. Therefore, CoV performance of direct estimation is much better than equation-based adaptation: the CoV of direct estimation is less than half of equation-based adaptation, when the measurement period is larger than 1 second.

Note that there is a trade-off between the smoothness and the response time, such that the estimated TCP throughput is getting smoother as the measurement period increases. However, long measurement period leads to large response time to converge to a steady state throughput.

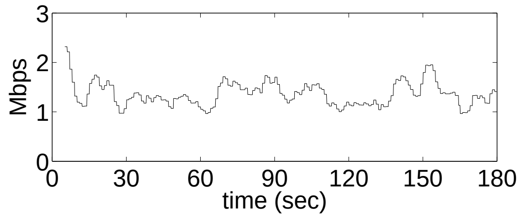
We can also find that the buffer occupancy in Figure 48 exhibits more predictable behavior. The variability of buffer occupancy is limited within 1Mbytes around the steady state buffer size, hence we can manage a playout buffer more easily.

5.4 Video Streaming Experiments

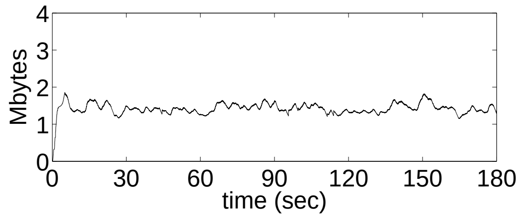
In this section, we show experimental results of video streaming over TCP to investigate the impact of the buffer size estimation algorithm and the throughput adaptation algorithm on video quality.

5.4.1 Error Resilient Codec

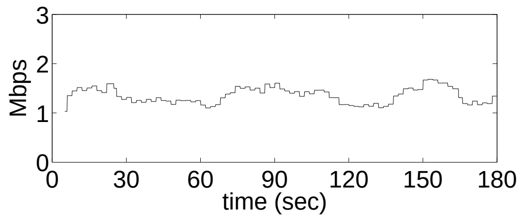
In the experiments, we generate a video sequence by digitizing an analog video followed by an error-resilient scalable codec proposed in [71]. The codec provides good error resilience and high compression scalability by generating independently decodable packets of the same priority. The codec is developed based on 3-D subband coding, which is an extension of the conventional 2-D subband decomposition to spatio-temporal subbands. High scalability can



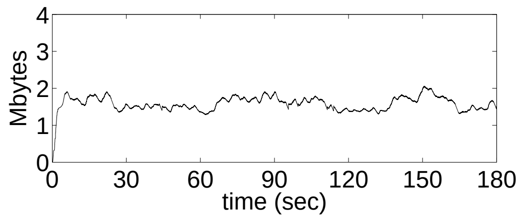
(a) Data transmission rate (T = 1s)



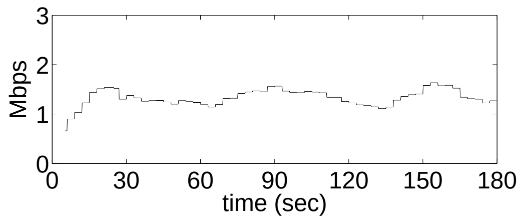
(b) Buffer occupancy (T = 1s)



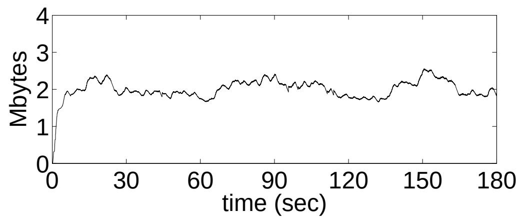
(c) Data transmission rate (T = 2s)



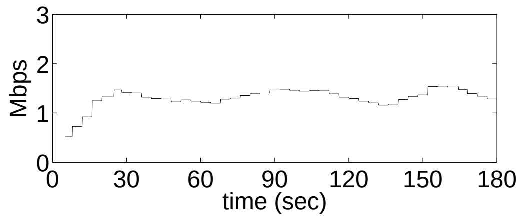
(d) Buffer occupancy (T = 2s)



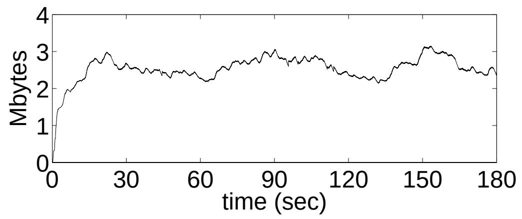
(e) Data transmission rate (T = 3s)



(f) Buffer occupancy (T = 3s)



(g) Data transmission rate (T = 4s)



(h) Buffer occupancy (T = 4s)

Figure 48: Performance of the direct estimation algorithm. The larger is the measurement period, the smaller is the data transmission rate variability.

be achieved by novel data partitioning in the subband coefficient domain. Specifically, each subband is partitioned into equal number of coefficient blocks, and packets are generated by grouping coefficient blocks from different subbands. Therefore, each packet has equal priority and can be decoded independently. Note that, since the codec has a constant number of bits per GOP, a CBR video stream can be generated easily (e.g., by allocating the same number of bits to each coefficient block).

Figure 49 shows the characteristic of error resilience. By varying random packet loss rate from 0 to 20%, the performance is measured in mean square error (MSE) which measures the pixel-by-pixel differences between an original video and a decoded video. Observe that MSE is exponentially decreased as packet loss rate is decreased. This property is corresponding to the convexity characteristic of general rate distortion functions [5]. However, traditional video codecs are optimized at a given data rate: if the channel data rate drops and is lower that the video coding rate, the received video quality decreases abruptly and the rate distortion function is a staircase curve. On the other hand, the error resilient codec is optimized over a wide range of data rates and robust to packet losses.

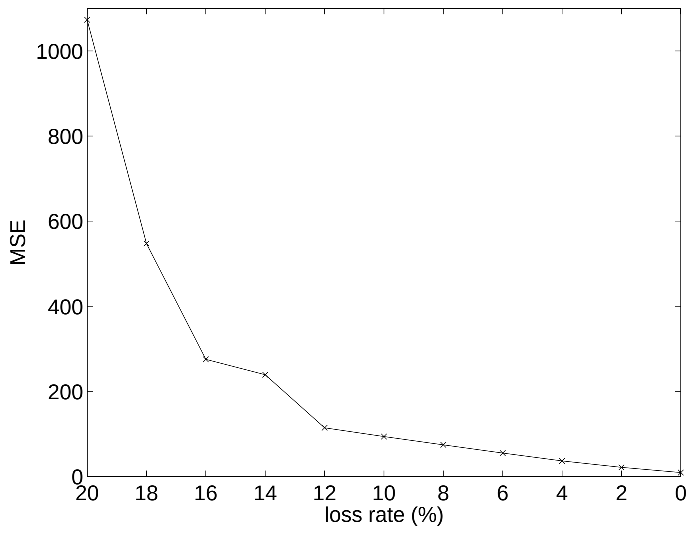


Figure 49: MSE characteristics of an error resilient codec. The codec exhibits the convexity characteristic to packet loss rate.

5.4.2 Performance Evaluation

In this section, we compare the quality of streaming video by measuring MSE. The video source is a 180-second scene of the *Harry Potter and the Sorcerer's Stone* movie encoded at 1.1Mbps. Specific encoding parameters are as follows: CIF resolution (352×288), 4 levels of spatial subband decomposition, and 3 levels of temporal subband decomposition (i.e., 8 frames/GOP). The encoded video is transmitted over the network model in Section 5.2.

We first show the performance of TCP streaming without receiver buffering in Figure 50. In this scenario, a receiver tries to display video as soon as media packets are received. Since the nature of AIMD and retransmission timeouts of TCP incurs a lot of variability, MSE of decoded video also fluctuates significantly and video quality is degraded.

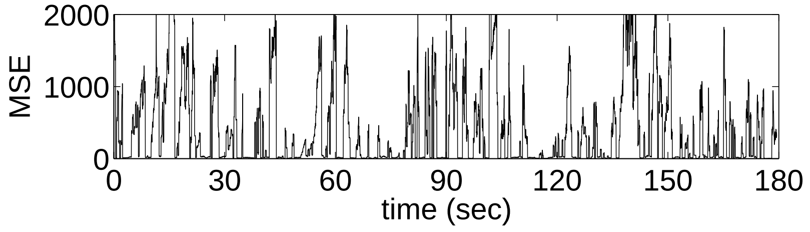
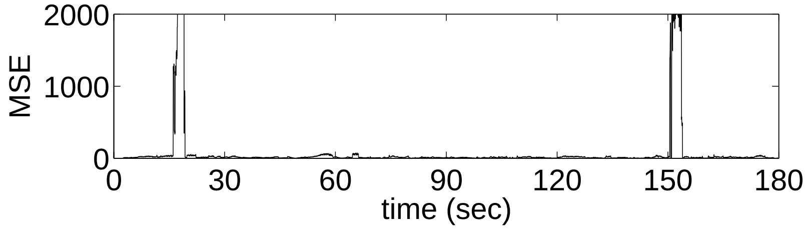


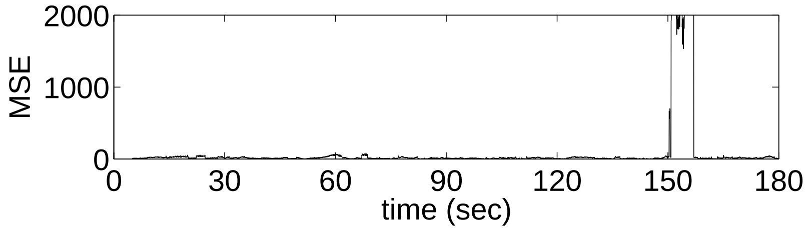
Figure 50: MSE of TCP without buffering. TCP streaming without receiver buffering incurs significant quality variability.

The impacts of receiver buffering on video quality are compared in Figure 51 and Figure 52. Figure 51 shows the performance of TCP streaming *without* any throughput adaptation algorithm, but the buffer size estimation algorithm is used with $60\widehat{R}$ estimation period. In the figures, abrupt MSE increases are corresponding to playout disruptions: a receiver stops displaying video and starts buffering received packets. Experimental results show that, as the buffering delay is increased, the number of playout disruptions is decreased, such that there are 2 disruptions in Figure 51 (a) and 1 in Figure 51 (b).

Note that this characteristic implies a trade-off between the video quality and waiting time: a user has to wait longer to achieve better quality. However, since a long waiting time is annoying and degrades user-perceived video quality, it is not desirable to set the buffer underrun probability too small (i.e., too large buffering delay). Therefore, we need the buffer size estimation algorithm to find an optimal waiting time.



(a) Buffer size estimation with $P_u = 4\%$

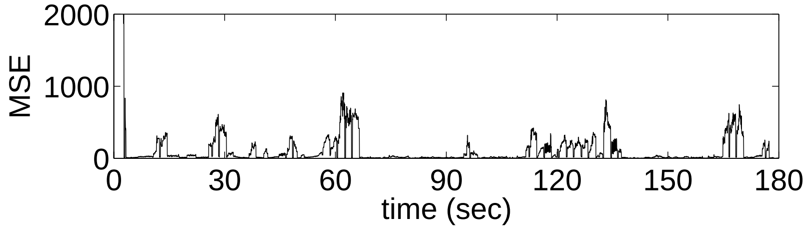


(b) Buffer size estimation with $P_u = 2\%$

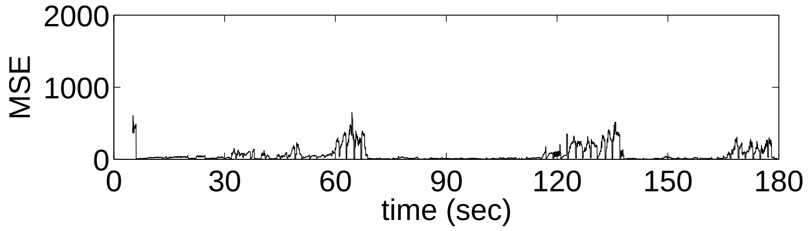
Figure 51: MSE performance of the buffer size estimation algorithm without throughput adaptation. As desired buffer underrun probability is decreased, measured buffer underrun probability is also decreased.

Figure 52 shows the performance of TCP buffering using the direct estimation algorithm as well as the buffer size estimation algorithm. In Figure 52 (a) and (b), the measurement periods of the direct estimation algorithm are set to 1.0 and 2.0 seconds, respectively. The estimation period of the playout buffer size estimation algorithm is set to $60\widehat{R}$. Experimental results demonstrate that realtime throughput adaptation used with an error resilient video codec can further improve the quality of received video: the decoded video does not show any playout disruption, although a few frames in decoded video exhibit noticeable distortions. It is also observed that the average MSE is decreased, as the buffer underrun probability is decreased. Compared with Figure 51, the direct estimation algorithm provides better perceived video quality, since decoded video is displayed without any disruption. This is not surprising, since the throughput adaptation algorithm exploits the error resilient characteristic of a player.

Similar observation was reported in [71]. The authors proposed a TCP friendly transport



(a) Buffer size estimation with $P_u = 4\%$



(b) Buffer size estimation with $P_u = 2\%$

Figure 52: MSE performance with the direct estimation algorithm. As desired buffer underrun probability is decreased, the average MSE is decreased.

protocol and streamed an encoded video over the protocol. Experimental results demonstrated the superiority of the protocol over TCP. However, their goal was to achieve low latency, such that overall delay is the sum of propagation delay and decoding time (i.e., there is no buffering delay). Hence, they did not investigate the effect of receiver buffering.

5.5 Conclusion

In this research, we propose realtime estimation algorithms to accommodate the rate variability of TCP in video streaming applications. We first develop a realtime estimation algorithm of playout buffer size based on the desired buffer size relationship. Since the Internet does not provide any information on network characteristic evolution, we estimate the RTT and the packet loss rate information, and compute the required buffer size and buffering delay. We also develop realtime throughput adaptation algorithms to prevent playback disruptions proactively. Experimental results show that the realtime buffer size estimation algorithm improves video quality of streaming video significantly, when used with a realtime throughput adaptation algorithm.

CHAPTER 6

CONCLUSION

Recent advances in video coding and the significant increase in network bandwidth have led to a number of video streaming applications. However, several technical challenges still remain. These include the heterogeneity of receivers, the fluctuation of available bandwidth, and the choice of transport protocols. In this thesis, we have investigated these challenges and developed techniques that maximize received video quality. The primary contributions of the research are summarized as follows:

- **A comparison of heterogeneous video multicast schemes** in which we have investigated three approaches that have been proposed to accommodate receiver heterogeneity in video multicast: 1) multicasting replicated video streams at different data rates, 2) multicasting cumulatively layered video in multiple layers, and 3) multicasting non-cumulatively layered video. While these protocols have been studied extensively before, there has been little in the way of fundamental understandings of how they can be compared from the point of view of network utilization and received video quality. We have developed a framework that can be used to evaluate these techniques and then showed results from applying this framework to various video multicasting scenarios. Experimental results showed that a fair comparison needs to take into account 1) the layering bandwidth overhead, 2) the specifics of encoding, 3) the protocol complexity to join and leave the appropriate multicast groups, and 4) the topological placement of receivers.

- **Optimal quality adaptation for scalable encoded video** in which we have considered the quality adaptation problem for scalable encoded VBR video. Since both the encoded video and the available bandwidth exhibit multiple time-scale variability, we have developed an optimal quality adaptation algorithm to minimize quality variability while increasing the usage of the available bandwidth. We also developed a realtime adaptation algorithm that does not require any information on the available bandwidth evolution. Experimental results showed that the quality adaptation algorithms reduce the quality variability significantly.

- **Dimensioning playout buffer requirements in video streaming over TCP** in which we have considered the use of TCP for video streaming applications. While the use of TCP provides reliable video stream delivery, the *sawtooth* behavior of TCP requires buffering at the receiver for smooth video playout. In current practice, however, there are no guidelines for the provisioning of this playout buffer, and smooth playout is insured through over-provisioning. The focus of this research is on memory-constrained clients where it is desirable to provide a buffer of just the right size to insure smooth playout while maintaining a prescribed video playout quality. We have derived analytic expressions for the minimum buffering requirements and then used this result to build workable realtime algorithms for the management of client-side buffering. Experimental results validated the buffering requirement relationships and showed that realtime adaptation algorithms improves video quality significantly.

APPENDIX A

MORE ON THE INFORMATION THEORETIC LIMITS

Consider a lossy data compression code consisting of one base layer and one enhancement layer. Let $\{X_i\}_{i=1}^{\infty}$ be a $\mathcal{X}$ valued discrete memoryless source with probability mass function, P, and $\mathcal{Y}_1$ be a finite reproduction alphabet. d_1 is defined by a non-negative valued mapping indicating a distortion measure, such that $d_1 : \mathcal{X} \times \mathcal{Y}_1 \to \mathbf{R}^+$.

Given a positive real value Δ_1 specifying the expected distortion, a rate distortion function $R(P, \Delta_1)$ that characterize the minimum achievable rate for a non-layered code is given by

$$R(P, \Delta_1) = \inf_{\substack{P_X = P, \\ \mathbf{E}d_1(X,Y_1) \leq \Delta_1}} I(X; Y_1),$$

where $I(X; Y_1)$ is the mutual information.

In the same way, the rate distortion function can be extended to the layered coding. Let $\mathcal{Y}_2$ be a finite reproduction alphabet and $d_2 : \mathcal{X} \times \mathcal{Y}_2 \to \mathbf{R}^+$ be a non-negative valued mapping representing a distortion measure. Suppose that a coding is done in the base layer with the rate $R_1 \geq R(P, \Delta_1)$ and the distortion Δ_2 is given, the minimum achievable rate in the enhancement layer is given by

$$R(P, R_1, \Delta_1, \Delta_2) = \inf_{\substack{P_X = P, \\ \mathbf{E}d_1(X,Y_1) \leq \Delta_1, \\ \mathbf{E}d_2(X,Y_2) \leq \Delta_2, \\ I(X;Y_1) \leq R_1}} I(X; Y_1 Y_2),$$

where $0 < \Delta_2 < \Delta_1$.

By definition, the rate distortion for a layered code is no smaller than that of a non-layered code. Hence, the following relationship generally holds [62].

$$R(P, \Delta_2) \leq R(P, R_1, \Delta_1, \Delta_2). \tag{18}$$

The condition under which equality holds requires that X, Y_1, and Y_2 satisfy a Markov condition [21]:

$$P_{XY_1Y_2}(x, y_1, y_2) = P_X(x)P_{Y_2|X}(y_2|x)P_{Y_1|Y_2}(y_1|y_2),$$

where $x \in \mathcal{X}, y_1 \in \mathcal{Y}_1$, and $y_2 \in \mathcal{Y}_2$.

From (18), the minimum data rate of non-layered stream is equal or smaller than that of layered stream. However, it is difficult to design a layered code satisfying Markov condition in general.

Moreover, it was shown that, even if the Markov condition holds, the performance of layered coding is inferior to that of non-layered coding in terms of the error exponent measure. Specifically, for n length block code, the error function is defined by the probability that the source, X^n, cannot be reproduced within distortion Δ_1 in the base layer or within distortion Δ_2 in the enhancement layer. It was shown in [35] that the error function converges to zero exponentially with the rate of the error exponent and that the error exponent for layered coding does not exceed that for non-layered coding.

APPENDIX B

PROOF OF THEOREM 1

We employ a continuous-time model to prove Theorem 1. Our notations for the continuous version of the variables are given in Figure 53.

$x_i(t)$	VOP size of the ith layer at time t
$X_i(t)$	cumulative data requirement: $X_i(t) = \int_0^t x_i(s)ds$
$s_i(t)$	selected data of the ith layer at time t: $s_i(t) = \begin{cases} x_i(t), & \text{for the } \texttt{select} \text{ state} \\ 0, & \text{otherwise} \end{cases}$
$S_i(t)$	cumulative selected data: $S_i(t) = \int_0^t s_i(u)du$
b_i	receiver buffer size to store unplayed ith layer video
$r_i(t)$	available bandwidth
$R_i(t)$	cumulative bandwidth constraint: $R_i(t) = \int_0^t r_i(s)ds$
$C_i(t)$	cumulative capacity of the ith layer: $C_i(t) = \min\{S_i(t) + b_i, R_i(t)\}$
$T_i(t)$	size of transmitted data
$U_i(t)$	network bandwidth utilization: $U_i(t) = \frac{T_i(t)}{R_i(t)}$

Figure 53: Notations for the continuous-time model

We first present a condition that achieves minimum quality variability. It is evident that the variability increases, as the number of transitions between the `select` state and the `discard` state increases during a fixed interval. Hence, long average sojourn time reduces average number of state transitions which leads to less variability. Our task is to determine a threshold in the quality adaptation framework to maximize average sojourn time.

Assume that we begin with enough network bandwidth, but state transition occurs at t_0, since $S_i(t_0) = C_i(t_0)$ as shown in Figure 54 (recall that the feasibility requires $C_i(t)$ to exceed $S_i(t)$). Let u be the sojourn time in the `discard` state. Then, a sender remains in the `discard` state until $t_0 + u$, such that $s_i(\eta) = 0$ for $t_0 \leq \eta < t_0 + u$. Given a threshold

d, the sender returns to the `select` state at time $t_0 + u$, since

$$C_i(t_0 + u) - S_i(t_0 + u) = d \tag{19}$$

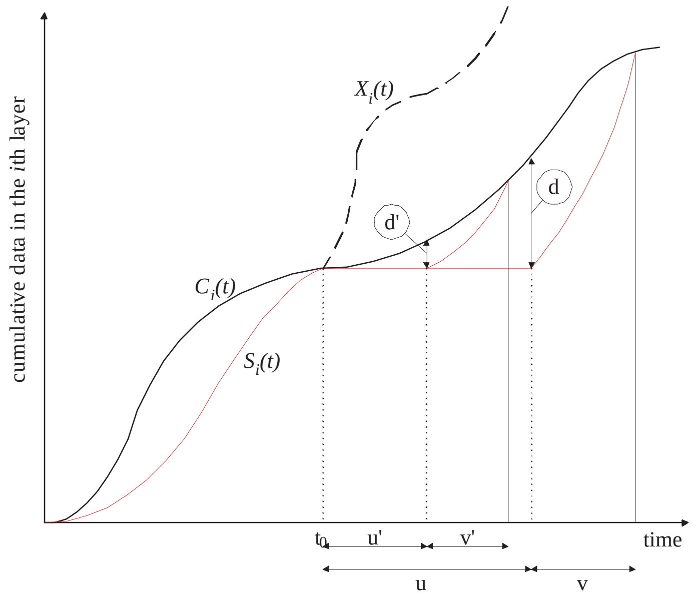


Figure 54: Illustration of quality adaptation

Let v be the sojourn time in the `select` state. With the threshold value d, we need to find the average sojourn time in the select state $E\{v\}$. The sender is assumed to enter the select state at t_0+u and to leave the state at t_0+u+v since $S_i(t_0+u+v) = C_i(t_0+u+v)$. Using (19),

$$\begin{aligned}
\int_{t_0+u}^{t_0+u+v} x_i(\eta)d\eta &= \int_{t_0+u}^{t_0+u+v} s_i(\eta)d\eta \\
&= \int_{t_0+u}^{t_0+u+v} s_i(\eta)d\eta + C_i(t_0+u+v) - S_i(t_0+u+v) \\
&= C_i(t_0+u) - S_i(t_0+u) + \int_{t_0+u}^{t_0+u+v} c_i(\eta)d\eta \\
&= d + \int_{t_0+u}^{t_0+u+v} c_i(\eta)d\eta.
\end{aligned}$$

If we assume $x_i(t)$, $c_i(t)$, and v are independent, conditional average yields $E\{v\}E\{x_i(t)\} = d + E\{v\}E\{c_i(t)\}$. Hence,

$$E\{v\} = \frac{d}{E\{x_i(t)\} - E\{c_i(t)\}}. \tag{20}$$

Consider a different scheme with a threshold of d', such that $d' < d$. Let u' be the sojourn time in the `discard` state, and v' be the sojourn time in the `select` state. From

(19) and (20),

$$\begin{aligned} C_i(t_0+u') &= S_i(t_0+u')+d' \\ E\{v'\} &= \tfrac{d'}{E\{x_i(t)\}-E\{c_i(t)\}} \end{aligned}$$

Since $S_i(t_0+u) = S_i(t_0+u')$ and $C_i(t)$ is non-decreasing,

$$\begin{gathered} u \geq u' \\ E\{v\} > E\{v'\}. \end{gathered} \tag{21}$$

From (21), as we increase the threshold value, average sojourn time spent in each state gets longer and we can achieve less variability. Therefore, the optimal threshold that achieves minimum quality variability is given by

$$\tau = \max_{\tau_i}\{\tau_i \mid \tau_i \in T, T \text{ is a set of feasible thresholds}\}. \tag{22}$$

Now we consider a threshold value that maximizes the network bandwidth utilization. Since the maximum sending rate is bounded by both the network bandwidth and the receiver buffer size, $T_i(t) \leq C_i(t) = \min\{S_i(t)+b_i, R_i(t)\}$. Therefore, utilization at time t is bounded by

$$\begin{aligned} U_i(t) &= \tfrac{T_i(t)}{R_i(t)} \\ &\leq \tfrac{C_i(t)}{R_i(t)} \\ &= \tfrac{\min\{S_i(t)+b_i, R_i(t)\}}{R_i(t)} \\ &= \begin{cases} \tfrac{S_i(t)+b_i}{R_i(t)}, & \text{if } S_i(t)+b_i < R_i(t) \\ 1, & \text{otherwise.} \end{cases} \end{aligned}$$

The necessary condition of $U_i(t) = 1$ is given by $S_i(t)+b_i \geq R_i(t)$ for all t. Since $C_i(t) = \min\{S_i(t)+b_i, R_i(t)\}$,

$$\begin{aligned} b_i &\geq R_i(t) - S_i(t) \\ &\geq C_i(t) - S_i(t). \end{aligned} \tag{23}$$

From (19), equation (23) implies

$$b_i \geq \tau. \tag{24}$$

Equation (24) shows that the threshold value τ should be equal or less than the receiver buffer size to achieve maximum bandwidth utilization.

From (22) and (24), the optimal threshold value is given by the receiver buffer size, i.e.,

$$\tau = b_i.$$

APPENDIX C

THREE-LAYER EXTENSION OF DYNAMIC THRESHOLD STREAMING

$$\begin{cases}
\pi_b[k] = 1, \pi_{FGS}[k] = 0, & \text{if } Y_b[k] < q_b(s), \\
\pi_b[k] = \alpha_b, \pi_{FGS}[k] = 1 - \alpha_b, & \text{if } Y_b[k] \geq q_b(s), Y_{FGS}[k] < q_{FGS}(s), \\
\pi_b[k] = \alpha_b, \pi_{FGS}[k] = \alpha_{FGS}, & \text{if } Y_b[k] \geq q_b(s), Y_{FGS}[k] \geq q_{FGS}(s), \\
\pi_b[k] = \alpha_b, \pi_{FGS}[k] = 0, & \text{if } Y_b[k] \geq q_b(s), Y_{FGS}[k] > \sum_{j=k}^{N} x_{FGS}[j], \\
\pi_b[k] = 0, \pi_{FGS} = 1, & \text{if } Y_b[k] > \sum_{j=k}^{N} x_b[j], Y_{FGS}[k] < q_{FGS}(s), \\
\pi_b[k] = 0, \pi_{FGS}[k] = \alpha_{FGS}, & \text{if } Y_b[k] > \sum_{j=k}^{N} x_b[j], Y_{FGS}[k] \geq q_{FGS}(s), \\
\pi_b[k] = 0, \pi_{FGS}[k] = 0, & \text{if } Y_b[k] > \sum_{j=k}^{N} x_b[j], Y_{FGS}[k] > \sum_{j=k}^{N} x_{FGS}[j],
\end{cases}$$

where $\pi_b[k]$, $\pi_{FGS}[k]$, and $\pi_{FGST}[k]$ are fraction of transmission resources; $x_b[k]$ and $x_{FGS}[k]$ are encoded data rate; $Y_b[k]$ and $Y_{FGS}[k]$ are buffer occupancy. The threshold value at time s is determined by $q_i(s) = C(r_i - \alpha_i X_{avg}(s))$, where $X_{avg}(s)$ is the estimated available bandwidth. r_i is the data rate of each layer, α_i is defined by $\alpha_i = \frac{r_i}{\sum r_i}$, C is set to 1 second, and $i = b, FGS, FGST$. Note that $\pi_b[k] + \pi_{FGS}[k] + \pi_{FGST}[k] = 1$.

APPENDIX D

PROOFS OF BUFFER SIZE REQUIREMENTS

D.1 Proof of (6)

To prove (6), we use the following relationships[1] [10]:

$$W_i(p) = \frac{W_{i-1}(p)}{2} + \frac{X_i}{b} - 1 \tag{25}$$

$$Y_i = \alpha_i + W_i(p) - 1 \tag{26}$$

$$Y_i = \frac{X_i}{2}[\frac{W_{i-1}(p)}{2} + W_i(p)] + \beta_i. \tag{27}$$

Note that the expressions in (25) and (27) are different from the original equations in [55] by a constant term[2]. However, for small values of p, TCP throughput in a TDP can still be expressed by

$$B_{TDP}(p, R) = \frac{1}{R}\sqrt{\frac{3}{2bp}} + o(\frac{1}{\sqrt{p}}). \tag{28}$$

To achieve a desired buffer underrun probability, we need to consider the expression of the minimum buffer size in (4) and (5) in Section 4.2.1. Using the Markovian inequality, the buffer underrun probability at time t in a TDP is given by

$$\begin{aligned} &P\{q_{min} \le 0 | t \in Z^{TD}\} \\ &\quad= P\{q_0 \le \frac{b}{8}[W_{i-1}(p) - 2\widehat{\lambda}(p) - 1]^2\} \\ &\quad\le \frac{E\{\frac{b}{8}[W_{i-1}(p) - 2\widehat{\lambda}(p) - 1]^2\}}{q_0} \\ &\quad= \frac{b}{8q_0}[E\{W^2\} - 4\widehat{\lambda}(p)E\{W\} + 4\widehat{\lambda}^2(p) + 4\widehat{\lambda}(p) - 2E\{W\} + 1], \end{aligned} \tag{29}$$

where $E\{W^2\}$ stands for the average of $W_{i-1}^2(p)$ and $E\{W\}$ for the average of $W_{i-1}(p)$.

[1]Although three duplicated packets are lost and not delivered to a receiver in a TDP, the data reception rate can be approximated by the data transmission rate for small p.

[2]The relationships can be verified from the ith TDP in Fig. 37 (a): the parameters given by $W_{i-1}(p) = W_i(p) = 6$, $X_i = 8$, $Y_i = 39$, $\alpha_i = 34$, $\beta_i = 3$, and $b = 2$ satisfy (25), (26), and (27).

Equation (29) can be solved using (25), (26), and (27). From (25), it follows that

$$E\{X\} = \frac{b}{2}E\{W\} + b. \tag{30}$$

Observe that squaring (25) leads to $W_i^2(p) = \frac{W_{i-1}(p)^2}{4} + \frac{W_{i-1}(p)X_i}{b} + \frac{X_i^2}{b^2} - \frac{2X_i}{b} - W_{i-1}(p) + 1$. Hence, the average of X_i^2 is given by

$$E\{X^2\} = \frac{3b^2}{4}E\{W^2\} - \frac{b^2}{2}E^2\{W\} + b^2E\{W\} + b^2. \tag{31}$$

Note that squaring (25) after manipulating the $\frac{W_{i-1}(p)}{2}$ term yields

$$W_i^2(p) - W_i(p)W_{i-1}(p) + \frac{W_{i-1}(p)^2}{4} = \frac{X_i^2}{b^2} - \frac{2X_i}{b} + 1. \tag{32}$$

From (30), (31), and (32), the correlation of congestion window sizes between adjacent TDPs is given by

$$E\{W_i(p)W_{i-1}(p)\} = \frac{1}{2}[E\{W^2\} + E^2\{W\}]. \tag{33}$$

We consider (26) and (27) to derive $E\{W^2\}$. Since α_i is the first packet lost in a TDP, α_i can be assumed to have a geometric distribution with the probability p. Hence, it follows that $E\{\alpha_i\} = \frac{1}{p}$ and $E\{\alpha_i^2\} = \frac{2-p}{p^2}$. With this assumption, squaring (26) leads to

$$E\{Y^2\} = \frac{2-p}{p^2} + E\{W^2\} + 1 + \frac{2}{p}E\{W\} - \frac{2}{p} - 2E\{W\}. \tag{34}$$

In the same way, $E\{Y^2\}$ can also be obtained by (27) and (33). Since β_i is the number of packets in the last round, it can be assumed to have a uniform distribution in $[1, W_i]$. Therefore, squaring (27) yields

$$\begin{aligned} E\{Y^2\} &= \frac{E\{X^2\}}{4}[\frac{5E\{W^2\}}{4} + E\{W_i(p)W_{i-1}(p)\}] + E\{\beta\}E\{X\}\frac{3E\{W\}}{2} + E\{\beta^2\} \\ &= \frac{b^2}{4}[\frac{3}{4}E\{W^2\} - \frac{1}{2}E^2\{W\} + E\{W\} + 1] \\ &\times [\frac{7}{4}E\{W^2\} + \frac{1}{2}E^2\{W\}] + \frac{3b}{4}E^2\{W\}[\frac{1}{2}E\{W\} + 1] \\ &+ \frac{2E\{W^2\} + 3E\{W\} + 1}{6}. \end{aligned} \tag{35}$$

Since the average window size is given by $E\{W\} = \sqrt{\frac{8}{3bp}} + o(\frac{1}{\sqrt{p}})$ in [55], we assume $E\{W^2\} = O(\frac{1}{p})$. By equating the relationships in (34) and (35), we can derive a relationship, such that

$$\frac{21b^2}{64}E^2\{W^2\} - \frac{b}{3p}E\{W^2\} - \frac{22}{9p^2} = o(\frac{1}{p^2}).$$

Hence,

$$E\{W^2\} = \frac{32+8\sqrt{478}}{63bp} + o(\frac{1}{p}). \tag{36}$$

Note that the long-term average of $\lambda_k(p)$ is equal to $\widehat{\lambda}(p)$. Therefore, TCP throughput in (28) can also be applied to $\widehat{\lambda}(p)$, such that $\widehat{\lambda}(p) = \sqrt{\frac{3}{2bp}} + o(\frac{1}{\sqrt{p}})$. From (36), the buffer underrun probability in a TDP in (29) is bounded by

$$\begin{aligned} P\{q_{min} \le 0 | t \in Z^{TD}\} &\le \frac{b}{8q_0}[\frac{32+8\sqrt{478}}{63bp} - 4\sqrt{\frac{3}{2bp}}\sqrt{\frac{8}{3bp}} + 4(\sqrt{\frac{3}{2bp}})^2] + o(\frac{1}{p}) \\ &= \frac{0.16}{q_0 p} + o(\frac{1}{p}). \end{aligned} \tag{37}$$

To derive an expression of the buffer underrun probability in a TOP, we consider (5) and the Markovian inequality. Consequently, we have

$$\begin{aligned} P\{q_{min} \le 0 | t \in Z^{TO}\} &= P\{q_0 \le \frac{Z^{TO}}{R}\widehat{\lambda}(p)\} \\ &\le \frac{E\{Z^{TO}\}}{q_0 R}\widehat{\lambda}(p), \end{aligned} \tag{38}$$

Since the average duration of a TOP is described by $E\{Z^{TO}\} = T_0\frac{f(p)}{1-p}$, where $f(p) = 1+p+2p^2+4p^3+8p^4+16p^5+32p^6$, (38) leads to

$$P\{q_{min} \le 0 | t \in Z^{TO}\} \le \frac{T_0}{q_0 R}\sqrt{\frac{3}{2bp}} + o(\frac{1}{\sqrt{p}}). \tag{39}$$

Now we derive the unconditional probability of buffer underrun using the conditional probabilities. Since $W_i(p)$ is a regenerative process over the period of $Z^{TD} + Z^{TO}$, we have

$$\begin{aligned} P\{t \in Z^{TD}\} &= \frac{E\{Z^{TD}\}}{E\{Z^{TD}\} + E\{Z^{TO}\}}, \\ P\{t \in Z^{TO}\} &= \frac{E\{Z^{TO}\}}{E\{Z^{TD}\} + E\{Z^{TO}\}}, \end{aligned}$$

where $E\{Z^{TD}\} = E\{n\}(E\{X\}+1)R$, and $E\{n\}$ is the average number of TDPs in Z^{TD}. Consider the probability of TO loss indication Q, which is given by $Q = \frac{1}{E\{n\}} \approx \min(1, 3\sqrt{\frac{3bp}{8}})$ in [55]. From (37), (39), the buffer underrun probability is thus

$$\begin{aligned} P\{q_{min} \le 0\} &= \frac{P\{q_{min} \le 0 | t \in Z^{TD}\}(E\{X\}+1)}{(E\{X\}+1) + QE\{Z^{TO}\}} + \frac{QP\{q_{min} \le 0 | t \in Z^{TO}\}E\{Z^{TO}\}}{(E\{X\}+1) + QE\{Z^{TO}\}} \\ &\le \frac{\frac{0.16}{q_0 p}R\sqrt{\frac{2b}{3p}} + \frac{T_0}{q_0 R}\sqrt{\frac{3}{2bp}}\min(1, 3\sqrt{\frac{3bp}{8}})T_0\frac{f(p)}{1-p}}{R\sqrt{\frac{2b}{3p}} + \min(1, 3\sqrt{\frac{3bp}{8}})\frac{T_0}{q_0 R}} + o(\frac{1}{p}) \\ &\le \frac{0.16}{q_0 p}[1 + \frac{9.4}{b}(\frac{T_0}{R})^2 \min(1, 3\sqrt{\frac{3bp}{8}})p(1+32p^2)] + o(\frac{1}{p}). \end{aligned}$$

Therefore, given desired buffer underrun probability, such that $P\{q_{min} \leq 0\} \leq P_u$, required buffer size is given by

$$q_0 \geq \frac{0.16}{pP_u}[1 + \frac{9.4}{b}(\frac{T_0}{R})^2 \min(1, 3\sqrt{\frac{3bp}{8}})p(1 + 32p^2)].$$

D.2 Proof of (9)

Since it is assumed that packets are drained faster than the arrival rate, required buffer size in the ith TDP is

$$q_0^i = q_0 - \sum_{j=1}^{i-1} X_j[\widehat{\lambda}(p) - \lambda_k(p)],$$

where q_0 is the required buffer size in TDP 0 (i.e., $q_0 = q_0^0$). The buffer underrun probability in TDP i is thus

$$\begin{aligned} P\{q_{min} \leq 0 | t \in TDP_i\} &= P\{q_0 - \sum_{j=1}^{i-1} X_j[\widehat{\lambda}(p) - \lambda_k(p)] - \frac{b}{8}[W_{i-1}(p) - 2\widehat{\lambda}(p) - 1)^2 \leq 0\} \\ &\leq \frac{b}{8q_0}E\{[W_{i-1}(p) - 2\widehat{\lambda}(p) - 1]^2\} + \frac{i-1}{q_0}E\{X_j(\widehat{\lambda}(p) - \lambda_k(p))\} \end{aligned} \tag{40}$$

Since the sum of the buffer underrun probability in all TDPs leads to the buffer underrun probability in a Z^{TD},

$$\begin{aligned} P\{q_{min} \leq 0 | t \in Z^{TD}\} &= \sum_i P\{q_{min} \leq 0 | t \in TDP_i\} P\{t \in TDP_i\} \\ &\leq \frac{b}{8q_0}E\{[W_{i-1}(p) - 2\widehat{\lambda}(p) - 1]^2\} \\ &+ \frac{E\{n\} - 1}{q_0}E\{X_j\}[\widehat{\lambda}(p) - E\{\lambda_k(p)\}], \end{aligned} \tag{41}$$

where $E\{n\}$ is the average number of TDPs in a Z^{TD}. Using the relationships $E\{n\} = 1/\min(1, 3\sqrt{\frac{3bp}{8}})$ and $E\{X_j\} = \sqrt{\frac{2b}{3p}}$, it follows that

$$\frac{E\{n\} - 1}{q_0}E\{X_j\}[\widehat{\lambda}(p) - E\{\lambda_k(p)\}] \approx \frac{\sqrt{\frac{2b}{3p}}(\widehat{\lambda}(p) - E\{\lambda_k(p)\})}{q_0 \min(1, 3\sqrt{\frac{3bp}{8}})}. \tag{42}$$

Therefore the buffer underrun probability is given by

$$P\{q_{min} \leq 0\} = \frac{P\{q_{min} \leq 0 | t \in Z^{TD}\}E\{Z^{TD}\}}{E\{Z^{TD}\} + E\{Z^{TO}\}} + \frac{P\{q_{min} \leq 0 | t \in Z^{TO}\}E\{Z^{TO}\}}{E\{Z^{TD}\} + E\{Z^{TO}\}}$$

$$
\begin{aligned}
&\leq \quad \frac{\frac{0.16}{q_0 p}R\sqrt{\frac{2b}{3p}} + \min(1, 3\sqrt{\frac{3bp}{8}})\frac{T_0}{q_0 R}\sqrt{\frac{3}{2bp}}T_0\frac{f(p)}{1-p}}{R\sqrt{\frac{2b}{3p}} + \min(1, 3\sqrt{\frac{3bp}{8}})T_0\frac{f(p)}{1-p}} + o(\frac{1}{p}) \\
&+ \quad \frac{\frac{\sqrt{\frac{2b}{3p}}(\widehat{\lambda}(p) - E\{\lambda_k(p)\})}{q_0\min(1, 3\sqrt{\frac{3bp}{8}})}R\sqrt{\frac{2b}{3p}} + \min(1, 3\sqrt{\frac{3bp}{8}})\frac{T_0}{q_0 R}\sqrt{\frac{3}{2bp}}T_0\frac{f(p)}{1-p}}{R\sqrt{\frac{2b}{3p}} + \min(1, 3\sqrt{\frac{3bp}{8}})T_0\frac{f(p)}{1-p}} \\
&\leq \quad \frac{0.16}{q_0 p}[1 + \frac{9.4}{b}(\frac{T_0}{R})^2\min(1, 3\sqrt{\frac{3bp}{8}})p(1+32p^2)] \\
&+ \quad \frac{\sqrt{\frac{2b}{3p}}(\widehat{\lambda}(p) - E\{\lambda_k(p)\})}{q_0\min(1, 3\sqrt{\frac{3bp}{8}})} + o(\frac{1}{p}). \qquad (43)
\end{aligned}
$$

To achieve $P\{q_{min} \leq 0\} \leq P_u$, required buffer size is

$$
q_0 \quad \geq \quad \frac{0.16}{pP_u}[1 + \frac{9.4}{b}(\frac{T_0}{R})^2\min(1, 3\sqrt{\frac{3bp}{8}})p(1+32p^2)] + \frac{\sqrt{\frac{2b}{3p}}(\widehat{\lambda}(p) - E\{\lambda_k(p)\})}{P_u\min(1, 3\sqrt{\frac{3bp}{8}})}. \quad (44)
$$

D.3 Proof of (10)

Since the window size in a TDP is limited by W_m, the receiver buffer size at round k is

$$
\begin{aligned}
q_k &= q_0 + \sum_{n=1}^{k}[\lambda_n(p) - \widehat{\lambda}(p)] \\
&= \begin{cases} \frac{k^2}{2b} + \frac{[W_m - 2\widehat{\lambda}(p) - 1]k}{2} + q_0, & \text{if } 0 < k \leq bW_m \\ W_m k - \frac{bW_m + 2\widehat{\lambda}(p)}{2} + q_0, & \text{otherwise,} \end{cases} \qquad (45)
\end{aligned}
$$

Assuming $\widehat{\lambda}(p)$ is equal to W_m, the Markovian inequality and the differentiation of (45) yield the buffer underrun probability in a TDP, such that

$$
\begin{aligned}
P\{q_{min} \leq 0 | t \in Z^{TD}\} &= P\{q_0 \leq \frac{b}{8}[W_m - 2\widehat{\lambda}(p) - 1]^2\} \\
&= P\{q_0 \leq \frac{b}{8}(W_m + 1)^2\} \\
&\leq \frac{b}{8q_0}(W_m + 1)^2. \qquad (46)
\end{aligned}
$$

In the same way, the buffer underrun probability in a TOP is given by

$$
\begin{aligned}
P\{q_{min} \leq 0 | t \in Z^{TO}\} &= P\{q_0 \leq \frac{Z^{TO}}{R}W_m\} \\
&\leq \frac{W_m}{q_0 R}E\{Z^{TO}\}. \qquad (47)
\end{aligned}
$$

Since $E\{X\} = \frac{b}{8}W_m + \frac{1-p}{pW_m} + 1$ and $E\{Z^{TO}\} = T_0\frac{f(p)}{1-p}$, the unconditional buffer underrun probability for small p is approximated by

$$\begin{aligned} P\{q_{min} \leq 0\} &\leq \frac{\frac{b}{8q_0}(W_m+1)^2(\frac{b}{8}W_m + \frac{1-p}{pW_m} + 2)R + \widehat{Q}(W_m)\frac{W_m}{q_0 R}[T_0\frac{f(p)}{1-p}]^2}{(\frac{b}{8}W_m + \frac{1-p}{pW_m} + 2)R + \widehat{Q}(W_m)T_0\frac{f(p)}{1-p}} \\ &\approx \frac{b}{8q_0}(W_m+1)^2. \end{aligned} \tag{48}$$

Given buffer underrun constraint $P\{q_{min} \leq 0\} \leq P_u$, required buffer size is thus

$$q_0 \geq \frac{b}{8P_u}(W_m+1)^2$$

REFERENCES

[1] J. Apostolopoulos, "Reliable video communication over lossy packet networks using multiple state encoding and path diversity," *VCIP 2001*, San Jose, CA, Jan. 2001.

[2] J. Apostolopoulos, T. Wong, W. Tan, and S. Wee, "On multiple description streaming with content delivery networks," *IEEE INFOCOM 2002*, New York, NY, June 2002.

[3] D. Bansal and H. Balakrishnan, "Binomial congestion control algorithms," *IEEE INFOCOM 2001*, Anchorage, AK, Apr. 2001.

[4] D. Bansal, H. Balakrishnan, S. Floyd, and S. Shenker, "Dynamic behavior of slowly-responsive congestion control algorithms," *ACM SIGCOMM 2001*, San Diego, CA, Aug. 2001.

[5] T. Berger, *Rate distortion theory: a mathematical basis for data compression*, Prentice-Hall, 1971.

[6] S. Bohacek, "A stochastic model of TCP and fair video transmission," *IEEE INFOCOM 2003*, San Francisco, CA, Apr. 2003.

[7] J.-C. Bolot and T. Turletti, "Adaptive error control for packet video in the Internet," *ICIP '96*, Lausanne, Switzerland, Sept. 1996.

[8] J.-C. Bolot, T. Turletti, and I. Wakeman, "Scalable feedback control for multicast video distribution in the Internet," *ACM SIGCOMM '94*, London, UK, Sept. 1994.

[9] J. Byers, M. Luby, and M. Mitzenmacher, "Fine-grained layered multicast," *IEEE INFOCOM 2001*, Anchorge, AK, Apr. 2001.

[10] Z. Chen, *Personal correspondence.*

[11] S. Y. Cheung, M. H. Ammar, and X. Li, "On the use of destination set grouping to improve fairness in multicast video distribution," *IEEE INFOCOM '96*, San Francisco, CA, Mar. 1996.

[12] Y.-H. Chu, S. G. Rao, and H. Zhang, "A case for end system multicast," *ACM SIGMETRICS 2000*, Santa Clara, CA, June 2000.

[13] *Coding of Audio-Visual Objects, Part-2 Visual, Amendment 4: Streaming Video Profile*, ISO/IEC 14496-2/FPDAM4, July 2000.

[14] Companion web site, `http://www.cc.gatech.edu/computing/Telecomm/people/Phd/tkim/qa.html`, 2003.

[15] Companion web site, `http://www.cc.gatech.edu/computing/Telecomm/people/Phd/tkim/tcp_streaming.html`, 2003.

[16] G. J. Conklin and S. S. Hemami, "A comparison of temporal scalability techniques," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, no. 6, pp. 909-919, Sept. 1999.

[17] T. Cover and A. El Gamel, “Achievable rates for multiple descriptions,” *IEEE Trans. Inform. Theory*, vol. 28, pp. 851-857, Nov. 1982.

[18] P. de Cuetos and K. W. Ross, “Adaptive rate control for streaming stored Fine-Grained Scalable video,” *NOSSDAV 2002*, Miami, FL, May 2002.

[19] P. de Cuetos, D. Saparilla, and K. W. Ross, “Adaptive streaming of stored video in a TCP-friendly context: multiple versions or multiple layers?” *Packet Video 2001*, Kyongju, Korea, Apr. 2001.

[20] P. Domański, A. Luczak, and S. Maćkowiak, “Scalable MPEG video coding with improved B-frame prediction,” *IEEE ISCAS 2000*, Geneva, Switzerland, May 2000.

[21] W. H. Equitz and T. M. Cover, “Successive refinement of information,” *IEEE Trans. Inform. Theory*, vol. 37, no. 2, pp. 269-275, Mar. 1991.

[22] Ethereal - network protocol analyzer, `http://www.ethereal.com/`.

[23] W. Fenner and S. Casner, “A traceroute facility for IP multicast,” *IETF draft, draft-ietf-idmr-traceroute-ipm-06.txt*, Mar. 2000.

[24] S. Floyd and K. Fall, “Promoting the use of end-to-end congestion control in the Internet,” *IEEE/ACM Trans. Networking*, vol. 7, no. 4, pp. 458-472, Aug. 1999.

[25] S. Floyd, M. Handley, J. Padhye, and J. Widmer, “Equation based congestion control for unicast applications,” *ACM SIGCOMM 2000*, Stockholm, Sweden, Aug. 2000.

[26] V. K. Goyal, J. Kočević, R. Arean, and M. Vetterli, “Multiple description transform coding of images,” *ICIP ’98*, Chicago, IL, Oct. 1998.

[27] M. Grossglauser, S. Keshav, and D. N. C. Tse, “RCBR: A simple and efficient service for multiple time-scale traffic,” *IEEE/ACM Trans. Networking*, vol. 5, pp. 741-755, Dec. 1997.

[28] Q. He, M. H. Ammar, G. Riley, and R. Fujimoto, “Exploiting the predictability of TCP’s steady-state behavior to speed up network simulation,” *IEEE/ACM MASCOTS 2002*, Fort Worth, TX, Oct. 2002.

[29] P.-H. Hsiao, H. T. Kung, and K.-S. Tan, “Video over TCP with receiver-based delay control,” *NOSSDAV 2001*, Port Jefferson, NY, June 2001.

[30] ISO/IEC, “Generic coding of moving pictures and associated audio information,” *ISO/IEC 13818-2*, 1995.

[31] ISO/IEC, “Coding of audio-visual objects – part 2: Visual,” *ISO/IEC 14496-2*, 1999.

[32] ITU, “Video coding for low bit rate communication,” *ITU-T Recommendation H.263*, 1998.

[33] V. Jacobson, “Congestion avoidance and control,” *ACM SIGCOMM ’88*, Stanford, CA, Aug. 1988.

[34] T. Jiang, M. H. Ammar, and E. W. Zegura, “Inter-receiver fairness: a novel performance measure for multicast ABR sessions,” *ACM SIGMETRICS ’98*, Madison, WI, June 1998.

[35] A. Kanlis and P. Narayan, “Error exponents for successive refinement by partitioning,” *IEEE Trans. Inform. Theory*, vol. 42, no. 1, pp. 275-282, Jan. 1996.

[36] F. Kelly, “Charging and rate control for elastic traffic,” *European Transactions on Telecommunications*, vol. 8, pp. 33-37, 1997.

[37] B.-J. Kim, Z. Xiong, and W. A. Pearlman, “Low bit-rate scalable video coding with 3D set partitioning in hierarchical trees,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, pp. 1374-1387, Dec. 2000.

[38] T. Kim and M. H. Ammar, “A comparison of layering and stream replication video multicast schemes,” *NOSSDAV 2001*, Port Jefferson, NY, June 2001.

[39] T. Kim and M. H. Ammar, “A comparison of heterogeneous video multicast schemes: layered encoding or stream replication?” *to appear in IEEE Transactions on Multimedia.*

[40] T. Kim and M. H. Ammar, “Optimal quality adaptation for MPEG-4 fine-grained scalable video,” *IEEE INFOCOM 2003*, San Francisco, CA, Mar. 2003.

[41] T. Kim and M. H. Ammar, “Optimal quality adaptation for scalable encoded video,” *to appear in IEEE Journal on Selected Areas in Communications.*

[42] T. Kim and M. H. Ammar, “Determining playout buffer requirements for video streaming over TCP to memory-constrained devices,” to be published.

[43] J.-I. Kimura, F. A. Tobagi, J. M. Pulido, and P. J. Emstad, “Perceived quality and bandwidth characterization of layered MPEG-2 video encoding,” *SPIE International Symposium Voice, Video, and Data Communications*, Boston, MA, Sept. 1999.

[44] L. P. Kondi, F. Ishtiaq, and A. K. Katsaggelos, “On video SNR scalability,” *ICIP ’98*, Chicago, IL, Oct. 1998.

[45] C. Krasic, K. Li, and J. Walpole, “The case for streaming multimedia with TCP,” *iDMS 2001*, Lancaster, UK, Sept. 2001.

[46] J. Y. Le Boudec and P. Thiran, “A short tutorial on network calculus I: fundamental bounds in communication networks,” *ISCAS 2000*, Geneva, Switzerland, May 2000.

[47] W. Li, “Overview of fine granularity scalability in MPEG-4 video standard,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 3, pp. 301-317, Mar. 2001.

[48] M. Li, M. Claypool, and R. Kinicki, “MediaPlayer versus RealPlayer - a comparison of network turbulance,” *ACM SIGCOMM IMW 2002*, Marseille, France, Nov. 2002.

[49] X. Li, S. Paul, and M. H. Ammar, “Layered video multicast with retransmission (LVMR): Evaluation of hierarchical rate control,” *IEEE INFOCOM ’98*, San Francisco, CA, Mar. 1998.

[50] S. Low, F. Paganini, and J. Doyle, “Internet congestion control,” *IEEE Control Systems Magazine*, vol. 22, no. 1, pp. 28-43, 2002.

[51] S. McCanne, V. Jacobson, and M. Vetterli, “Receiver driven layered multicast,” *ACM SIGCOMM ’96*, Stanford, CA, Aug. 1996.

[52] Microsoft windows media player, `http://www.microsoft.com/windows/windowsmedia/9series/player.aspx`.

[53] S. Nelakuditi, R. R. Harinath, E. Kusmierek, and Z.-L. Zhang, "Providing smoother quality layered video stream," *NOSSDAV 2000*, Chapel Hill, NC, June 2000.

[54] `NS-2` network simulator, `http://www.isi.edu/nsnam/ns/`, 2001.

[55] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP reno performance: a simple model and its empirical validation," *IEEE/ACM Trans. Networking*, vol. 8, no. 2, pp. 133-145, Apr. 2000.

[56] V. Paxson, "End-to-end Internet packet dynamics," *IEEE/ACM Trans. Networking*, vol. 7, no. 3, pp. 277-292, June 1999.

[57] J. Postel, "Transmission control protocol specification," *RFC-793*, Menro Park, CA, Sept. 1981.

[58] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numberical recipes in C*, Cambridge university press, 1988.

[59] H. M. Radha, M. van der Schaar, and Y. Chen, "The MPEG-4 fine-grained scalable video coding method for multimedia streaming over IP," *IEEE Trans. Multimedia*, vol. 3, no. 1, pp. 53-68, Mar. 2001.

[60] A. Reibman, H. Jafarkhani, Y. Wang, M. Orchard, and R. Puri, "Multiple description coding for video using motion compensated prediction," *ICIP '99*, Kobe, Japan, Oct. 1999.

[61] R. Rejaie, D. Estrin, and M. Handley, "Quality adaptation for congestion controlled video playback over the Internet," *ACM SIGCOMM '99*, Cambridge, MA, Aug. 1999.

[62] B. Rimoldi, "Successive refinement of information: Characterization of the achievable rates," *IEEE Trans. Inform. Theory*, vol. 40, no. 1, pp. 253-259, Jan. 1994.

[63] A. Said and W. A. Pearlman, "An embedded wavelet hierarchical image coder," *IEEE ICASSP '92*, San Francisco, Mar. 1992.

[64] A. Said and W. A. Pearlman, "A new fast and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, pp. 243-250, June 1996.

[65] J. D. Salehi, Z.-L. Zhang, J. Kurose, and D. Towsley, "Supporting stored video: reducing rate variability and end-to-end resource requirements through optimal smoothing," *IEEE/ACM Trans. Networking*, vol. 6, no. 4, pp. 397-410, Aug. 1998.

[66] D. Saparilla and K. W. Ross, "Optimal streaming of video," *IEEE INFOCOM 2000*, Chapel Hill, NC, June 2000.

[67] D. Saparilla and K. W. Ross, "Streaming stored continuous media over fair-share bandwidth," *NOSSDAV 2000*, Tel Aviv, Israel, Mar. 2000.

[68] N. Shacham, "Multipoint communication by hierarchically encoded data," *IEEE INFOCOM '92*, Florence, Italy, 1992.

[69] W. Tan, W. Cui, and J. G. Apostolopoulos, "Playback buffer equalization for streaming media using stateless transport prioritization," *Packet Video 2003*, Nantes, Fr, Apr. 2003.

[70] W. Tan and A. Zakhor, "Video multicast using layered FEC and scalable compression," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 3, pp. 373-386, Mar. 2001.

[71] W. Tan and A. Zakhor, "Real-time Internet video using error resilient scalable compression and TCP-friendly transport protocol," *IEEE Trans. Multimedia*, vol. 1, no. 2, pp. 172-186, June 1999.

[72] V. Vaishampayan and S. John, "Interframe balanced multiple description video compression," *ICIP '99*, Kobe, Japan, Oct. 1999.

[73] G. Voronov and M. Feder, "The redundancy of successive refinement codes and codes with side information," *ISIT 2000*, Sorrento, Italy, June 2000.

[74] Y. Wang, M. Claypool, and Z. Zuo, "An empirical study of realvideo performance across the Internet," *ACM SIGCOMM IMW 2001*, San Francisco, CA, Nov. 2001.

[75] W. Willinger, M. S. Taqqu, R. Sherman, and D. V. Wilson, "Self-similarity through high-variability: Statistical analysis of Ethernet LAN traffic at the source level," *ACM SIGCOMM '95*, Cambridge, MA, Aug. 1995.

[76] Y. R. Yang, M. S. Kim, and S. S. Lam, "Optimal partitioning of multicast receivers," *ICNP 2000*, Osaka, Japan, Nov. 2000.

[77] E. W. Zegura, K. L. Calvert, and M. J. Donahoo, "A quantitative comparison of graph-based models for internetworks," *IEEE Trans. on Networking*, vol. 5, no. 6, pp. 770-783, Dec. 1997.

[78] Y. Zhang, V. Paxson, and S. Shenker, "The stationarity of internet path properties: Routing, loss, and throughput," *ACIRI technical report*, May 2000.

[79] C. Zhang and V. Tsaoussidis, "TCP-Real: improving real-time capabilities of TCP over heterogeneous networks," *NOSSDAV 2001*, Port Jefferson, NY, June 2001.