

Scalable Application-Aware Router Mechanisms

A Thesis
Presented to
The Academic Faculty

by


Ashraf A. Awad

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


School of Electrical and Computer Engineering
Georgia Institute of Technology
June 2003

Scalable Application-Aware Router Mechanisms

Approved by:



Professor Martin W. McKinnon, Advisor



Professor John Copeland

Professor Chuanyi Ji

Professor Raghupathy Sivakumar, Co-Advisor

Date Approved 7/16/2003

To Mom, Dad

and

Peace In The World.

ACKNOWLEDGEMENTS

I extend my sincere gratitude and appreciation to many people who made this Ph.D. thesis possible. Special thanks to my Advisors Professor Martin McKinnon and Professor Raghupathy Sivakumar for their assistance, guidance and support during my study.

I am deeply thankful to my family for their support and encouragement during my study. I would like to give my special thanks my brothers Naser and Anwar for their continuing encouragement, advice and support.

I am blessed with great friends that I met during the past years. I would like to express my deepest gratitude to my dear friends for the good times and the good laughs that we shared together.

TABLE OF CONTENTS

| | |
|---------------------------------------------------------|------|
| DEDICATION | iii |
| ACKNOWLEDGEMENTS | iv |
| LIST OF FIGURES | viii |
| CHAPTERS | |
| APPENDICES | |
| SUMMARY | x |
| 1 INTRODUCTION | 1 |
| 2 UTILITY AWARENESS IN ROUTER MECHANISMS | 7 |
| 2.1 Example: MPEG-2 Video | 8 |
| 2.2 MPEG-2 Structure and Traffic Models | 9 |
| 2.3 Packet Loss in Video Traffic | 10 |
| 2.4 Utility Metric | 11 |
| 3 BACKGROUND AND RELATED WORK | 13 |
| 3.1 Buffer Management Schemes | 13 |
| 3.2 Video Loss Estimation | 15 |
| 3.3 Utility-Based Rate Allocation | 16 |
| 4 GOODPUT EVALUATION AND ESTIMATION | 19 |
| 4.1 Introduction | 19 |
| 4.2 Overview | 21 |
| 4.2.1 Network Model and Assumptions | 21 |
| 4.2.2 Goals | 22 |
| 4.3 Queueing Analysis | 23 |
| 4.3.1 Queueing Model Description | 23 |
| 4.3.2 Markov Chain Definition | 24 |
| 4.3.3 $R(t, w)$ Matrices Description | 26 |
| 4.3.4 $C(t, w, x)$ Structure | 28 |
| 4.3.5 $D_{ij}(t, w, x, z)$ Matrix Description | 29 |

| | | |
|----------|---------------------------------------------------------|-----------|
| 4.4 | Details of Matrix Description | 31 |
| 4.4.1 | C_a Matrix Description | 31 |
| 4.4.2 | C_n Matrix Description | 35 |
| 4.5 | PBS-D Queueing Analysis | 36 |
| 4.6 | Analysis of Other Schemes | 37 |
| 4.6.1 | TBS Queueing Analysis | 37 |
| 4.6.2 | TBS-D Queueing Analysis | 38 |
| 4.6.3 | TD and TD-D Queueing Analysis | 39 |
| 4.7 | Simulation and Results | 39 |
| 4.7.1 | Simulation Environment | 39 |
| 4.7.2 | Video Traffic Model | 40 |
| 4.7.3 | Buffer Management Schemes | 42 |
| 4.7.4 | Effect of Changing Load and Buffer Size | 45 |
| 4.8 | Case Study: Admission Control | 47 |
| 4.8.1 | Admission Control Failure | 47 |
| 4.8.2 | Bandwidth Over-provisioning | 49 |
| 4.9 | Issues and Discussion | 50 |
| 5 | MPFD: UTILITY AWARE BUFFER MANAGEMENT | 55 |
| 5.1 | Introduction | 55 |
| 5.2 | Overview | 57 |
| 5.3 | MPFD Algorithm | 59 |
| 5.4 | MPFD with Jitter and Frame Loss | 62 |
| 5.4.1 | Network Jitter | 62 |
| 5.4.2 | Frame Loss | 66 |
| 5.5 | Deployment Considerations | 67 |
| 5.6 | Simulation and Results | 69 |
| 5.6.1 | Metrics | 69 |
| 5.6.2 | Results | 70 |
| 6 | UTILITE: UTILITY AWARE RATE ALLOCATION | 76 |
| 6.1 | Introduction | 76 |

| | | |
|-------|-----------------------------------------------------------|-----|
| 6.2 | Utility-Based Fairness | 78 |
| 6.2.1 | Utility Functions | 78 |
| 6.2.2 | Motivation | 79 |
| 6.2.3 | Utility fairness | 81 |
| 6.3 | Ideal Utility Max-Min Rate Allocation Algorithm | 82 |
| 6.3.1 | Ideal Algorithm | 83 |
| 6.3.2 | Packet Drop | 84 |
| 6.4 | Overview of Utilite | 84 |
| 6.4.1 | Utilite Framework | 84 |
| 6.4.2 | Challenges | 86 |
| 6.4.3 | Design Elements | 88 |
| 6.4.4 | Overhead Evaluation | 90 |
| 6.5 | Utilite Design | 91 |
| 6.6 | Simulation and Results | 93 |
| 6.6.1 | Simulation Environment | 93 |
| 6.6.2 | Results | 93 |
| 6.7 | Issues and Discussion | 98 |
| 7 | CONCLUSIONS AND FUTURE WORK | 104 |
| | APPENDIX A — TBS QUEUEING ANALYSIS | 105 |
| | APPENDIX B — TBS-D QUEUEING ANALYSIS | 108 |
| | REFERENCES | 111 |

LIST OF FIGURES

| | | |
|----|----------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 1 | QoS Dimensions | 8 |
| 2 | (a) GOP structure, (b) GOP display order, (c) GOP transmission order. . . | 9 |
| 3 | Comparison between Goodput and raw packet loss. | 12 |
| 4 | Slotted queueing system. | 23 |
| 5 | (a) ON-OFF source pattern. (b) Effect of buffer overflow in a B-Frame (frame drop). (c) Effect of buffer overflow in an anchor frame (GOP drop). | 24 |
| 6 | Pessimistic event order within a slot. | 24 |
| 7 | Queueing System Description. | 40 |
| 8 | Markov Chain video traffic model. | 40 |
| 9 | PBS goodput-based CLP in anchor frames. | 41 |
| 10 | PBS goodput-based CLP in B-frames. | 42 |
| 11 | PBS total goodput-based CLP. | 43 |
| 12 | Bounded lognormal frame size distribution for video source models. | 44 |
| 13 | TBS goodput-based CLP in anchor frames. | 45 |
| 14 | TBS goodput-based CLP in B-frames. | 46 |
| 15 | TBS total goodput-based CLP. | 47 |
| 16 | PBS-D goodput-based CLP in anchor frames. | 48 |
| 17 | PBS-D goodput-based CLP in B-frames. | 49 |
| 18 | PBS-D total goodput-based CLP. | 50 |
| 19 | TBS-D goodput-based CLP in anchor frames. | 51 |
| 20 | TBS-D goodput-based CLP in B-frames. | 52 |
| 21 | TBS-D total goodput-based CLP. | 53 |
| 22 | Effect of buffer size on goodput-based CLP. | 54 |
| 23 | Effect of load on goodput-based CLP. | 54 |
| 24 | MPFD virtual buffer compared to POB physical buffer. | 56 |
| 25 | GOP frames assignments. | 60 |
| 26 | MPFD algorithm | 63 |
| 27 | Passing interarrival information to past frames. | 64 |
| 28 | MPFD-JC algorithm | 65 |

| | | |
|----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 29 | MPFD-LC algorithm | 67 |
| 30 | Total loss. | 69 |
| 31 | Total loss for 20% load. | 70 |
| 32 | Loss in B-frames for 20% load. | 71 |
| 33 | Loss in I-frames for 20% load. | 72 |
| 34 | Loss in P-frames for 20% load. | 73 |
| 35 | Total loss for 23% load. | 74 |
| 36 | Loss in B-frames for 23% load. | 74 |
| 37 | Loss in I-frames for 23% load. | 75 |
| 38 | Loss in P-frames for 23% load. | 75 |
| 39 | Utility corresponding to rate-based rate allocation | 78 |
| 40 | Utility corresponding to rate-based rate allocation | 79 |
| 41 | Utility corresponding to rate-based rate allocation | 79 |
| 42 | Utility corresponding to rate-based rate allocation | 80 |
| 43 | Utility function approximation showing some of the parameters carried in the Utilite header. | 91 |
| 44 | Network topology. | 93 |
| 45 | Utility functions used in simulation. | 94 |
| 46 | Fair utility for flow with the same utility function. | 95 |
| 47 | Rate for flows with the same utility functions. | 96 |
| 48 | Utilite scalability with the number of different utility functions. Ten flows are used in all cases but the number of different utility functions varies. In figures:(d),(e), and (f), aggregate link rate and individual flow rates are shown. | 97 |
| 49 | Fair utility at the first congested link with two different utility functions. . | 98 |
| 50 | Fair utility at the second congested link with two different utility functions. | 99 |
| 51 | Rate at the second congested link with two different utility functions. . . . | 100 |
| 52 | Fair utility change with some flows with limited rate. | 101 |
| 53 | Rate change with some flows with limited rate. | 101 |
| 54 | Rate Change for dynamic flows. | 102 |
| 55 | Rate Change for dynamic flows. | 102 |
| 56 | Fair utility change for dynamic flows. | 103 |

SUMMARY

As network applications become more diverse, the requirements that they place on the network infrastructure become more varied and more stringent, particularly with respect to Quality of Service (QoS) requirements. Given this requirement, the network will intuitively yield more desirable service if its nodes have as much knowledge as feasible concerning the requirements of the applications whose traffic is traversing them. This goal can be achieved through mechanisms such as rate allocation, buffer management, congestion control, etc. As a large number of applications are served concurrently in the network, the fruits of this “application awareness” in the network will not be reaped unless it is efficiently scalable with respect to the number of traversing applications.

The objective of this thesis is to investigate “scalable application-aware” router mechanisms. This objective is addressed by discussing two types of mechanisms: buffer management and rate allocation. First, the queueing performance for several existing buffer management schemes is estimated and evaluated using Markov chains (in addition to simulation) with goodput as the metric instead of packet loss. Then, a new buffer management scheme is proposed to combine the efficiency and scalability extracted from existing schemes. The performance of the new scheme is compared with that of the existing schemes.

Second, a new scalable utility-based fair allocation architecture is proposed. This rate allocation architecture collects the utility functions from all flows that traverse a network link and uses them to allocate bandwidth such that all applications have similar utility (or application-level QoS). This is achieved without per-flow state, thereby achieving our goal of scalability for this thesis.

CHAPTER 1

INTRODUCTION

There are several services that a typical network currently provides to the end users. These services are facilitated by several network router mechanisms that include routing, buffer management, and rate allocation. Most existing network mechanisms are based on best-effort service in handling network packets. As a result, these services do not differentiate between network applications that have special service requirements; we illustrate such behavior with the following examples of router mechanisms.

- Routing protocols such as RIP, OSPF, and BGP [25, 42, 47] mostly forward packets with respect to the packet destination address and do not efficiently differentiate between packet or application types in their forwarding decision. Packets that have special service requirements can be handled as part of the routing protocol's extensions when they are explicitly specified during routing packets to their destination.
- The most widely used buffer management scheme is “first-in-first-out (FIFO)”, in which packets are dropped when the network buffer overflows, regardless of the packet type and its importance to the user's application. Therefore, a high-priority packet could be dropped whilst it may be possible to drop a low-priority packet, potentially resulting in a higher quality of service (QoS) degradation experienced at the end user. While discarded packets can be recovered through retransmission, some applications such as real-time multimedia streaming, forgo retransmission due to stringent delay constraints ¹.
- There are several rate allocation schemes such as WFQ, DRR, FRED, CSFQ, and CoreLite [39, 45, 53, 55]; however, these mechanisms try to provide rate allocation according to the available rate and the number of flows sharing a link, but not according

¹Other buffer management schemes that can provide priority discard will be discussed in Chapter 3.

to how that rate will impact the application's QoS. Consider a flow with two traffic layers: a basic layer and an enhancement layer, for example. The basic layer is an independent layer and contains information that provides the session with a coarse (or minimum) quality. The enhancement layer, however, is used to provide a higher session quality only when coupled with the basic layer; this layer cannot be used when it encounters packet loss. When a flow with layered traffic is allocated more bandwidth than the rate of the basic layer, but not enough to entirely accommodate the rate of the enhancement layer, some packets will be dropped. When the enhancement layer suffers from packet loss, its successfully transmitted packets will not be used by the end user; moreover, these packets will still be consuming bandwidth and contributing to network congestion. In addition, congestion may cause packet loss for the basic layer, resulting in an unnecessarily severe QoS degradation. Packet loss in the basic layer can be minimized by labeling its packets as packets that need special handling and processing in the network entities. This could be handled at the network layer level in the Internet Protocol (IPv4) by using a header field called *type of service* (TOS), which can be used to differentiate packet types. This field, however, is not currently utilized. This network environment cannot provide bandwidth and delay guarantees to the network users; thus, it will be impossible for the network to provide QoS for the users.

With the fast growth in high-speed communications networks, however, the spectrum of network applications that require QoS has been enlarged to include new and diverse bandwidth-demanding and delay-sensitive applications such as video and audio streaming, interactive video conferencing and games, and medical applications, etc. Since these applications require QoS that best-effort service cannot guarantee, they have increasingly been affected by mechanisms that facilitate such simple network service. To explain this effect, we take MPEG-2 [41] video streaming as one example of network applications that can benefit from network QoS assurances. Video frames are delay constrained and frames cannot be used if they miss their assigned display time. Therefore, a rate allocation mechanism that is unaware of the bandwidth and delay requirements of the video stream may waste

bandwidth in transmitting useless packets that have already expired. Also, the MPEG-2 compressed video format contains dependencies in coding between frames, and losing a frame may result in several frames being incorrectly displayed because of their dependencies on the lost frame. Therefore, in a buffer management scheme that is not aware of the video characteristics, high-priority packets can be dropped instead of available and preferable low-priority packets, resulting in higher video quality degradation than necessary even though comparable packet losses are seen in either case.

These problems motivate our addressing the need for new network models that provide QoS guarantees required by contemporary applications. The QoS of these applications may be severely degraded if they are subject to best-effort service along with other aggressive, less QoS-sensitive applications.

Several proposed network models try to provide QoS for network applications, such as IntServ and DiffServ [6, 8, 43, 63]. The IntServ network model is based on reserving network resources at each node for network applications to meet their QoS requirements. The user can specify the QoS level it needs and request it from the network. Reservation protocols such as RSVP [10, 28, 62] are used for reserving network resources and maintaining the connection. IntServ faces scalability issues because a connection information state for each flow has to be maintained at each network node on the connection route, resulting in increasing the computational and resource cost at each node as the number of traversing flows increases.

DiffServ, on the other hand, is based on classifying the application flows into several classes. Each class is given a priority relative to the other classes. Flow classification occurs at the edge network nodes, while the core network is kept simple with a fixed number of states that correspond to the service classes it supports. Since no per-flow state is kept, this model is scalable. However, this model provides less stringent QoS guarantees than IntServ.

In both IntServ and DiffServ, the application needs to provide the network with several connection parameters for admission control. Therefore, the application needs to map its session requirements to the mandated network QoS model's specifications. However, the

session requirements may not always match the model's specifications, which will usually result in either network underutilization or QoS degradation. In MPEG-2, for example, the network can reserve the peak rate for the video flow using IntServ, but this will result in underutilization because of the bursty nature of the video traffic characteristics and the high peak-to-average rate ratio. On the other hand, reserving less bandwidth with less stringent guarantees, as in DiffServ, will result in randomly dropping packets during long traffic bursts, resulting in video quality degradation that may exceed expectations as a result of error propagation caused by the MPEG-2 coding methodology.

Another network model that can provide QoS assurances to network applications is active networks [12, 58]. In this model, a packet carries instructions that are interpreted by the network node and then initiate processes to handle that packet. This model is designed to run application-specific packet processes at the network core nodes. However, it involves a significant amount of overhead because router's processing resources need to be distributed between the processes managing and controlling the router operation itself and the processes that are initiated by the network packets. Moreover, running processes at the network core raises network security issues because of the risk of allowing malicious code to execute at the network router. Scalability is another limiting issue in active networks. When a packet arrives at the router, for example, a process will be initiated to handle that packet, which results in the significant overhead of process initiation and code execution; as routers currently accommodate on the order of a million simultaneous flows², the issue of scalability becomes evident.

The above network models provide diverse QoS assurances through several network QoS metrics such as bandwidth and packet loss. Network applications use these metrics to achieve a certain service (i.e. satisfaction) level desired, which is characterized by the application-level QoS or utility. Utility represents the perceived QoS at the end application³. This raises an important question: what is the relationship between the QoS that the network provides and the application-level QoS?

²One example of such routers is *Apeiro*TM that is a product of Caspian Networks Inc.

³Utility can be defined as a function of one or more network QoS metrics and will be explained later in this section

To understand the difference between the two QoS metrics, we consider the rate allocation mechanism that provides equal rates to flows traversing it. While this rate allocation mechanism can potentially achieve equitable sharing of network resources, it falls short of recognizing the benefit of the allocated bandwidth to the application-level QoS or flow utility. The bandwidth allocated to an application is positively perceived only when it translates into an improvement in the application-level QoS. Let us re-visit the rate allocation example with layered traffic in more detail. We consider a flow that carries layered traffic with a basic layer rate of 400kbps and an enhancement layer of 200kbps (e.g. RLM [26]). Each layer will be used at the end user only if it is completely received. If the flow share provided to that flow is 400kbps, the flow will be transmitting only the basic layer, and therefore, will be optimally utilizing its allocated bandwidth (since the traffic consuming the allocated bandwidth is usable). If the fair share at the router increases to 500kbps and the flow starts transmitting the enhancement layer in a total rate of 600kbps, the enhancement layer will suffer from packet drops because the flow rate exceeds the allocated bandwidth and whole layer will be dropped at the destination. Thus, despite the increased bandwidth allocated to the flow, the end application will not receive any quality improvement from the network and its utility remains unchanged. On the contrary, the excess bandwidth, 100kbps, is wasted at the network. A utility-aware (or application-aware⁴) rate allocation mechanism would give that flow either 400kbps or 600kbps. Therefore, it is important for the network to be utility-aware to successfully provide network users with application-acknowledged quality enhancements. Utility awareness is directly coupled with fairness because, while 600kbps may result in a complete satisfaction for the flow with layered traffic, it may result in only 50% or less satisfaction for another bandwidth-demanding application.

As the utility of a network application tend to degrade during network congestion, we focus our work on providing application-awareness in router mechanisms that regulate network resource usage during congestion. One of the main router components on the flow's path is buffer management that, if achieved efficiently, can greatly contribute to enhancing the flow's utility especially for flows that carry prioritized packets. A related and equally

⁴We interchangeably exchange utility- and application-awareness in this document.

important router component is rate allocation, which become necessary during congestion to manage the available network bandwidth to similarly serve all network flows. In addition to being utility-aware, it is important for router mechanisms to be scalable because they typically manage a large number of network flows [55].

In this work, we aim to balance scalability and the quality that the network provides by introducing, at the network nodes, efficient and scalable application-aware services that improve the session quality over existing network services. The main contributions of this thesis fall in two categories, buffer management and rate allocation.

- In buffer management, we propose a *new simple and efficient buffer management scheme that is application-aware* and achieves the performance of complex schemes. This scheme utilizes the characteristics of the application's priority structure when discarding a packet. Also, we present a *performance estimation methodology based on Markov chains* for MPEG-2 video traffic. The methodology depends on the coding structure of MPEG-2 video in evaluating the effect of packet loss on video quality, and we use the estimation methodology to evaluate several existing buffer management schemes.
- In rate allocation, we address *fair rate allocation with respect to the utility curves for each flow*. Utility-based fairness is based on providing all users equal utilities, irrespective of differences in the bandwidth they use. Considering its deployment in a core network router with a large number of flows, we address the utility-based rate allocation in a *scalable environment*.

The thesis is organized as follows. In Chapter 2 we discuss utility awareness in network mechanisms. In Chapter 3, we present a literature review about loss estimation, buffer management, and rate allocation. We present the video loss estimation methodology in Chapter 4. In Chapter 5, we present the new application-aware buffer management scheme. We address scalable utility based rate allocation in Chapter 6. Finally, we conclude the thesis in Chapter 7.

CHAPTER 2

UTILITY AWARENESS IN ROUTER MECHANISMS

The utility of an application i is often represented by a utility function, U_i . The utility function is a curve that represents how the service s_i provided by the network translates into the performance (or utility) of application i . The network service s_i represents one or many of the ways in which the network can affect the quality received by the applications. Examples of service metrics are packet loss, delay, jitter, bandwidth, etc. An increase in s_i may result in an improvement in the application utility due to better service given from the network. U_i is usually expressed as a function of one metric such as bandwidth or packet loss.

The utility of network applications react differently to the dynamic changes in the QoS provided by the network. We identify four popular network QoS dimensions that affect an application's utility: bandwidth, packet loss, delay, and jitter. The utility of applications may change differently as any of those QoS dimensions vary, as shown in Figure 1. For example, an application that can tolerate bandwidth oscillations, like FTP, has a smoothly changing utility with bandwidth, while one that requires rigid bandwidth assurances, like real-time video, has a sharp change in utility when its bandwidth requirements are violated. This results in different utility functions of bandwidth across applications. Another example is the packet loss, in particular the type of the dropped packet. Losing a high-priority packet has a more impact on the application utility than a low-priority packet. These examples show that the same network service s_i can be perceived differently by different users, and therefore, the network service metrics are not accurate measures for the general applications' satisfaction. For the network to provide applications with similar service at the application level, it is important for it to sense how the user perceives the QoS it provides. Having the network aware of the impact of its service on the application QoS can result in better management of the network resources, which, in turn, increases the end-to-end application

perceived quality.

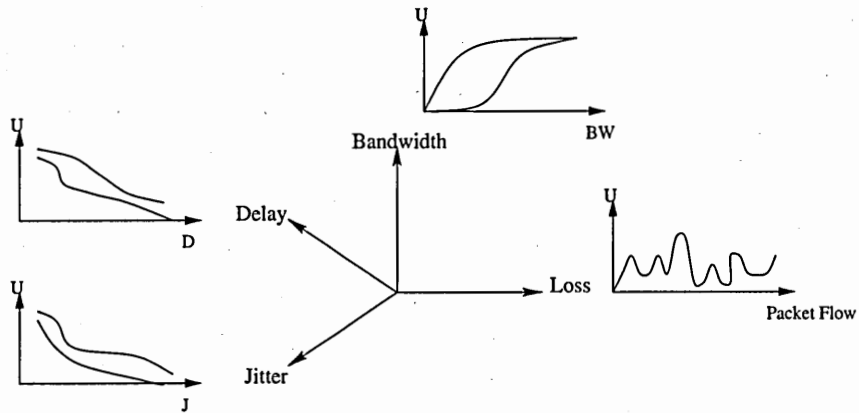


Figure 1: QoS Dimensions

To demonstrate the value of utility awareness in network mechanisms, we use buffer management as an example. Consider FIFO: this scheme is simple, scalable, and efficient when all packets arriving at the buffer are of the same type. However, this scheme causes performance degradation to applications that send packets of different priorities, particularly when the information carried in low-priority packets is dependent on that of preceding high-priority packets. Therefore, if a high-priority packet is lost, this may nullify the information in following low-priority packets even if they are received correctly at their destination. Since the FIFO scheme does not differentiate between high- and low-priority packets, it is likely that a high-priority packet is dropped when it would be preferable (by using a different buffer management scheme) to drop a low-priority packet, thereby degrading the overall QoS.

2.1 Example: MPEG-2 Video

Now we demonstrate the effect of not distinguishing between different packet priorities and their utility using an example of MPEG-2 video traffic. MPEG-2 is an efficient and popular compression standard that utilizes the spatial and temporal correlation between pictures in a video sequence. In the next section, we present an overview of MPEG-2 video structure.

2.2 MPEG-2 Structure and Traffic Models

Three types of frames are generated in MPEG-2 video-coded streams: *intra frames* (I-frames), *predicted frames* (P-frames), and *bidirectional frames* (B-frames). I-frames are coded using the information in the picture itself only, P-frames are coded with respect to the most recently preceding I-frame or P-frame (anchor frame), and B-frames are coded with respect to the most recently preceding and following anchor frame. I-frames are independent of other frames and have no temporal compression. P-frames are less compressed than B-frames (because their motion compensation coding is with respect to only one anchor frame), but they are usually smaller than the I-frames. B-frames are generally the smallest because their coding is dependent on the information contained in the two closest anchor frames through using motion compensation vectors.

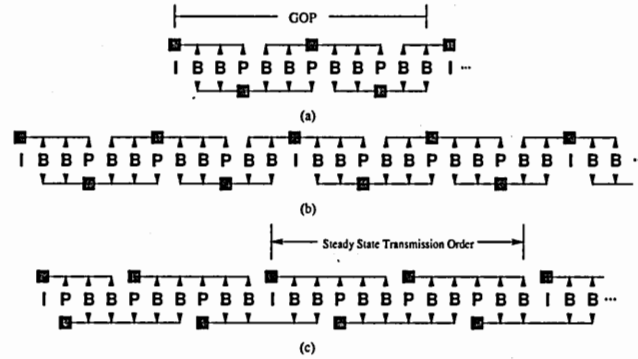


Figure 2: (a) GOP structure, (b) GOP display order, (c) GOP transmission order.

MPEG-2 frames are generated at a constant rate (e.g., 25 or 30 frames per second). They are generated in a regular, repeating sequence (see Figure 2(a)¹) called a *group of pictures* (GOP). Figures 2(b) and (c) give the order and dependencies between frames during both frame generation and display, and during transmission, respectively. The frames are ordered prior to transmission so that frame information is always transmitted before all other frames that are dependent upon it. In other words, when a frame arrives at the destination, the

¹The GOP pattern depends on the coder used to produce the MPEG-2 video stream. For example, some GOP patterns have higher number of B-frames such as *IBBBBPBBBBPBBBBPBBBB* or no B-frames at all, such as *IPPPIPPP*.

decoder will have all information needed from other frames in order to decode it. The reordering of frames creates an overlap between consecutive GOP boundaries.²

Several traffic models for MPEG video streams have been proposed in the literature. The models vary from capturing the short-range dependency (SRD) characteristics (e.g., Markov chains [1, 16, 21, 50] and AR models [1, 9, 11, 15, 21, 27, 29, 36, 40, 44, 65]), long-range dependency (LRD) characteristics (e.g., self-similar models [1, 21, 23, 48]), and models that attempt to approximate a probability density function (PDF) based on an experimentally acquired histogram [21, 35, 48]. Cell loss has been used as a measure to evaluate the agreement of proposed video traffic models with empirical data [18, 24, 34, 67]. Some of the common distributions used to model video traffic are lognormal and gamma distributions.

2.3 Packet Loss in Video Traffic

Because of the structure of MPEG-2 video stream, the impact of a lost (or corrupted) video packet will not only result in the affected frame being incorrectly displayed at the destination, but the impact of that error may also “ripple” through other subsequent frames because of the previously described interdependency. The propagation of the error will continue until the next synchronization point, (e.g., a picture, GOP, or sequence header). For example, losing an anchor frame will result in losing all the following frames in that GOP. In a buffer management scheme like FIFO that does not recognize the packet priority, losing a high-priority packet from an anchor frames will encounter higher penalty than that of a low-priority packet from a B-frame, which results in higher quality loss in the video stream.

To evaluate the performance of a buffer management scheme on an application, we need to evaluate its impact of the application’s quality. It is clear that the quality loss cannot be measured using packet loss ratio in the video stream since we need to account for packets that are received at the destination but are not usable. Therefore, we need to define a different metric that is more accurate and representable.

²Notice that the GOP structure in Figure 2 is an example that will be used in later discussions.

2.4 Utility Metric

To consider information interdependency between video frames, measuring video quality using cell³ loss as a metric would neither be suitable nor accurate because cell loss does not capture frame corruption caused by error propagation between frames. Hence, using frame goodput as a measurement metric for video quality is more adequate because it captures error propagation. We define *frame goodput* as the ratio between the cells in uncorrupted and correctly displayable frames to the total number of cells in the video stream that arrive to the buffer, as shown in equation (1). In addition, we define *goodput loss* as shown in equation (2). A similar definition of goodput was used in [38, 59, 66]. Notice that the goodput calculation is different from throughput calculation shown in equation (3) since the goodput excludes the unusable cells in its calculations in addition to the dropped cells.

The goodput can be studied at the slice⁴ level rather than at the frame level, but this will require complex analysis and difficult tracking of error propagation because of the possible slice interdependency on several slices from the previous frames. Even though this definition of the goodput over-estimates the loss in video quality, it is simpler to apply at network nodes than other more accurate definitions. Many other methods have been proposed for reducing the effect of cell loss in MPEG-2 streams through layered coding techniques [64] and error concealment [18], resulting in improved stream perceived quality.

$$\text{Goodput} = \frac{\text{No. of packets in good and usable frames}}{\text{Total No. of packets}} \quad (1)$$

$$\text{Goodput loss} = 1 - \text{Goodput} \quad (2)$$

$$\text{Throughput} = 1 - \frac{\text{No. of dropped packets}}{\text{total number of packets}} \quad (3)$$

³We use the term “cell” to refer to any fixed-size packet; therefore, a cell does not refer only to an ATM cell.

⁴A slice is a horizontal strip within a frame and it is the basic processing unit in the MPEG video coding scheme.

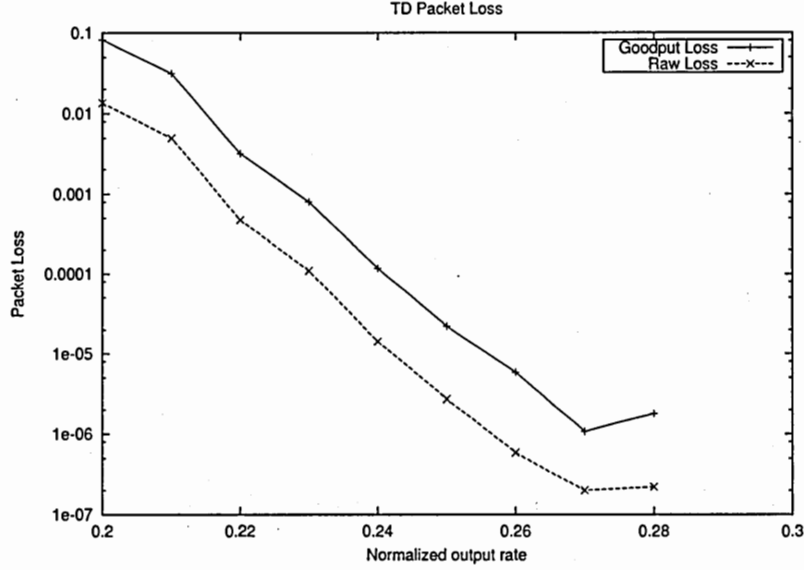


Figure 3: Comparison between Goodput and raw packet loss.

The difference between using goodput and raw packet loss as metrics in evaluating buffer management schemes is shown in Figure 3. This figure shows the packet loss in MPEG-2 video traffic for a buffer with Tail Discard (TD) dropping scheme (or FIFO). It is shown that the goodput loss is higher than the raw packet loss because goodput accounts for error propagation between packets. Hence, using the goodput as a utility metric presents a better measure of the usability of packets at the end user than raw packet loss.

Similar to buffer management mechanisms, rate allocation mechanisms that do not incorporate the application utility in its allocation criterion can wrongfully measure the QoS seen at the end user as discussed earlier in Chapter 1 where it was shown that an increment in the allocated bandwidth to a flow may not result in any improvement in the application utility. A similar evaluation can be performed to quantify the effect of utility awareness in rate allocation. We defer a detailed discussion of utility awareness to Chapter 6 for the interest of space.

CHAPTER 3

BACKGROUND AND RELATED WORK

Before we present our contributions in utility-aware buffer management and rate allocation schemes, we first look at several schemes in buffer management that are used later chapters in evaluating our proposed mechanisms. Then we present the related work from the contemporary literature wherein we address the ability of existing buffer management and rate allocation mechanisms to enhance the utility of network applications scalably with respect to the number of flows accommodated.

Several buffer management schemes have been studied in the literature wherein performance evaluation and analysis was provided. Although these schemes were analytically evaluated using raw packet loss as a metric, their performance has not been analytically studied or estimated using utility or goodput. Therefore, we also discuss the analytical evaluation of these schemes from the utility prospective.

3.1 Buffer Management Schemes

There are two approaches to minimizing the effect of packet loss in QoS-sensitive traffic, e.g. MPEG-2: (i) hop-by-hop approaches that addresses losses in intermediate nodes within the network, and, (ii) end-to-end approaches that may entail changing the encoding schemes of the traffic content such as error concealment and resilience algorithms. Although both approaches are important, we will focus in this work on the first approach and address it by examining different buffer management schemes where it is possible to minimize the unrecoverable errors that have the greatest impact on the perceived quality, e.g., losing an anchor frame in video streams.

Several studies of priority dropping schemes were found in the literature. Some of these schemes include partial buffer sharing (PBS), triggered buffer sharing (TBS), and push out buffer (POB).

In PBS, if the buffer occupancy is below a threshold T , both low- and high-priority packets are accepted into the buffer; otherwise only the high-priority packets are accepted until the buffer is full, when all arriving packets are discarded. In TBS, however, two thresholds are defined, $T_H > T_L$. When the buffer occupancy exceeds T_H , the buffer will not accept low-priority packets, and only high-priority packets are allowed into the buffer. The buffer goes back to accept all packets only when the buffer occupancy decreases below T_L . Thresholds in PBS and TBS are fixed in the buffer and do not change to accommodate burstiness in high-priority packets within a traffic flow or between flows that may be aggregated into the buffer. PBS and TBS are simple to implement and can provide buffer protection for high-priority frames, but the overall performance is low because the buffer is not fully utilized because of the fixed partial buffer space allocated to low-priority packets.

In POB, if a high-priority packet arrives at the buffer when it is full, a low-priority packet is removed from the buffer. If there are no low-priority packets in the buffer, the incoming packet is discarded. There are two versions of POB discussed in [17]: FIFO and LIFO. The difference between the two versions lies in the selection of low-priority packets to remove. In FIFO, when a low-priority packet is to be removed from the buffer, the packet closest to the head of the buffer is removed. In LIFO, however, the packet closest to the tail of the buffer is removed. While the POB scheme is efficient and fully utilizes the buffer space, it is complex to implement because it requires searching the buffer for low-priority packets to discard.

In [30], a queueing analysis for PBS was presented. The effect of load, burstiness, and buffer size had on packet loss probability was evaluated. In [17], Cuenca *et al.* presented a performance evaluation for PBS, TBS, and the two versions of POB. Real-time MPEG-2 traces were used in their evaluation. It was shown that TBS was the most effective in reducing the loss of anchor frames. The authors used packet loss rate as a performance measure, but did not consider whether or not the transmitted packets were usable at the receiver.

POB with FIFO and LIFO was studied in [59] for layered video. It was shown that

there was an optimum load at which the maximum rate of usable frames can be achieved. As the load increased beyond the optimum point, the rate of usable frames decreased. It should be noted that measuring usable frames rather than packet loss is a better measure of video quality as it accounts for the interdependency of frames. This frame rate can be increased if the dependency structure of the video is utilized in the dropping scheme itself. Zheng *et al.* in [66] proposed a scalable video transmission scheme that did not require major changes in network protocols. This scheme recognized some, but not all, of the frame dependencies in the video structure; the dependency between P-frames was ignored.

3.2 Video Loss Estimation

Analytical estimation techniques have many benefits in both the research and industry communities. While simulation techniques are beneficial and can provide the same results as estimation, analytical estimation techniques are often faster and can be applied in actual systems to support router mechanisms in call admission control, rate allocation, and buffer management schemes. These estimation techniques can also be used as a base for testing new performance improvement techniques. Markov chains were used in queueing analysis for VBR traffic models with multiple priorities [30, 32]. However, video traffic characteristics and the usefulness of the received packets were not recognized .

Some work in the literature has focused on queueing analysis and cell loss analysis for VBR and priority assignment network traffic; however, these studies did not consider the video traffic dependency structure, subsequent packet dropping, and deterministic modeling of video sequences. In [30, 32], Markov chains were used to analyze queueing mechanisms with focus on priority queueing or bursty VBR traffic; however, video traffic characteristics and usefulness of transmitted packets were not recognized. In [57], the impact of cell loss on the quality of video was studied; the loss analysis, though, simply depended on the probability of losing picture headers, GOP headers, and sequence headers in the video sequence. The authors did not consider all of the dependencies between pictures and did not verify their results with simulation. In [49], cell loss analysis for video traffic was presented using discrete time analysis; however, error propagation and dependency between frames

were not considered. None of the above work studied subsequent dropping of cells within a frame or frames within a GOP, which follows the dependency structure between frames in an MPEG-2 sequence. The authors in [38] studied the goodput of CBR video traffic and presented simple dropping algorithms. Their analysis of CBR traffic, however, differs from that for VBR traffic where the video frame sizes are variable. Their study provided neither a goodput analysis nor an estimation for the studied system.

In [14], an algorithm for fair bandwidth sharing in the core network without per-flow state was presented. Each flow was divided into sets of layers, and packets were marked at an edge router with a layer label (color) according to flows bit rates. When congestion occurs, the core router discards packets with the highest label number. That scheme was applied in evaluating the goodput of video traffic. However, it only served as a priority scheme since the core router did not explicitly discard video frames that reference the previously dropped frame during the congestion.

3.3 Utility-Based Rate Allocation

Because of the differences in resource requirements between network applications, an allocated bandwidth may be sufficient for one network application, but may not meet the minimum requirements for other applications. For example, for a certain bandwidth allocation, the utility of an FTP session may be excellent, but the utility for a real-time multimedia network application may be poor (because it requires specific latency and jitter guarantees).

At any network link with limited resources, these resources should be equivalently distributed among contending users according to a standard criteria. One possible criterion would entail equally distributing the available bandwidth among all contending users [55, 53]. This is a reasonable model when the cost of resources is uniform for all users; different sessions, however, may not achieve equivalent utilities for each of their end users. Another criterion may call for distributing bandwidth so that users sharing the link have equal utilities. In this case, the pricing model will be different because the network resources may not be equally distributed among the contending users.

In [22], the authors used utility curves to distribute bandwidth between wireless network users. FEC error protection was applied to packets to enable recovery from network errors. Rate allocation was based on the actual transmission rate including FEC overhead. The utility curves used were the same for all flows in the network, which resulted into a simple allocation model.

In [31], the authors proposed an algorithm to maximize the sum of all users' utilities in the network. In this scheme, depending on the feedback from the network¹, bandwidth is increased for each session in proportion to the derivative of the utility curve. When the network becomes congested, the rate of each flow is decreased by a number of predefined steps proportional to the number of congested links along the flow's path. The utility curves used for all sessions are $U(x_s) = w_s \ln x_s$. The modifications of the rate are carried out at the source. This approach, however, did not achieve similar utility assignment for all flows even though it might have increased the total network utility, especially when the utility functions differed.

In [60], the authors proposed a scalable algorithm to achieve utility-based rate allocation. In this scheme, each packet is labeled with an incremental utility that is equal to the slope of the flow's utility curve at the operating point. When a core router is congested, the packets with the lowest incremental utility are dropped. This resulted in maximizing the aggregate user utility. This algorithm, however, did not achieve provide similar utility to all users because flows with slowly increasing utility curves experienced more packet loss and less utility than other flows with rapidly increasing utility curves.

In [13], the authors proposed an algorithm to achieve a max-min utility fairness for all flows in the network. In this scheme, the utility functions for all flows are used to iteratively calculate the bandwidth that corresponds to the max-min fair utility for each flow in the network. A centralized algorithm was presented first to demonstrate the algorithm. Then, a distributed version of the algorithm was discussed. This distributed algorithm required maintaining a state for each flow in the network, which makes it less likely to be applied in

¹Network feedback is implemented by ACK messages that follow the same downstream path and carry the information about congested links.

intermediate network routers.

In [51], the authors proposed an algorithm for utility fairness in multi-rate multicast networks. They introduced the notion of “maximally fair rate allocation” for discrete bandwidth allocation, which was referred to as a weaker fairness model than max-min allocation. They proposed a polynomial complexity algorithm for computation of maximally fair rates allocated to various flows.

We consider a fairness approach to provide similar utilities among users in a congested link. Therefore, it is different from other approaches that try to maximize the sum of utilities or the total network utility. Our study addresses achieving utility fairness among users sharing a link in a scalable environment, which makes it different than schemes that achieve utility fairness.

CHAPTER 4

GOODPUT EVALUATION AND ESTIMATION

4.1 Introduction

Several buffer management schemes are evaluated in this chapter using goodput as the utility metric. We study and evaluate three buffer management schemes: TD, PBS and TBS. As part of the study, we also present an estimation algorithm for the goodput performance of these schemes using Markov chains. The estimation results are presented along with the simulation results to demonstrate the accuracy of estimation and to avoid any redundancies. The majority of this chapter is devoted to explain the analysis methodology applied in this estimation algorithm.

We use Markov chains to study and estimate the video quality deterioration resulting from congestion and buffer overflow at an access network node's queue using a variety of buffer management schemes¹. Providing an estimate to video quality has many benefits. Let us take rate call admission control as an example, when a connection is requested for a certain video traffic profile and an upper bound is specified for packet² loss, estimation can be used to approximate the needed bandwidth to meet the packet loss requirement. While simulation techniques are beneficial and can provide the same results as estimation, analytical estimation techniques are often faster and can be applied in actual systems to support applications like rate allocation, buffer management schemes, and call admission control mentioned above. It can also be used as a base for testing new performance improvement techniques.

As mentioned in Section 3.2, Markov chains have been used to estimate the queueing performance of video traffic. However, many of the estimated systems did not consider video traffic characteristics and the usefulness of the transmitted packets when the video quality

¹We explain in next section the reason behind choosing an access network node.

²We will interchangeably use the terms packets and cells to refer to fixed-size data units.

is measured, resulting in underestimates of the actual video quality loss [30, 32]. To consider the information dependency between video frames, measuring video quality using cell loss as a metric would neither be suitable nor accurate because cell loss does not capture frame corruption that is caused by error propagation between frames. It is important that the measured loss reflect the video quality reduction at the end user. *Hence, in our estimation methodology, we use a novel metric called “goodput”³ as the performance measure of video quality because it takes into account packets that can be successfully transmitted but not utilized at the destination, which happens if the video packet is dependent in its decoding on other packets that were discarded earlier.* The goodput metric has not been used with Markov chains previously for the evaluation and estimation of the video quality. We propose a method that is based on one presented in [54] for estimating the frame goodput of MPEG-2 video traffic buffered at the network buffer.

During congestion, video quality is primarily affected by the buffer management scheme used to drop packets. Simple network routers tend to use basic buffer management schemes (e.g., tail discard) or simple threshold-based schemes (e.g., partial buffer sharing [17]). These schemes, in general, are designed for Internet traffic and do not recognize video traffic properties. Smarter network routers have the capability to accommodate video traffic properties by applying more sophisticated buffer management schemes such as dependency dropping mechanisms. In dependency dropping, dropping a packet results in discarding other packets or frames that are dependent on the lost packet. Hence, the choice of the buffer management scheme during congestion determines the resulting video quality. *In this work, we consider several simple buffer management schemes suitable for simple routers, and smart buffer management schemes suitable for smart routers that can handle the added complexity.*

We segment video frames into fixed-size packets referred to as *cells*. This is similar to slotted systems like ATM and HFC networks. This segmentation is used because both VBR video frames and the slices that compose these frames vary in size. Therefore, measuring losses in terms of frames or slices would be misleading.

³The authors [38] used the goodput metric to measure the video quality but in a different context.

We apply the goodput estimation algorithm for three cases of existing buffer management schemes and compare the results with simulation results to compare their goodput performance and to verify the accuracy and flexibility of the estimation process. Then, we modify those schemes by adding a packet dropping mechanism that is guided by the dependency between video frames. The modified schemes will be referred to as “smart buffer management schemes”. In these schemes, dropping a packet will result in discarding other packets or frames that are dependent on the lost packet. These schemes are intended for network nodes that can implement sophisticated algorithms. Similarly, we apply the goodput estimation algorithm to these smart buffer management schemes and verify our results with simulation. We also show how the Markov chain is modified for each scheme. The three schemes are: tail dropping (TD), partial buffer sharing (PBS), and triggered buffer sharing (TBS). Through out this chapter, these schemes will be denoted as simple schemes.

The smart buffer management schemes are denoted as TD-D, PBS-D, and TBS-D, which correspond to TD, PBS, and TBS, respectively. In these smart schemes, the buffer becomes aware of the video frame dependency structure by discarding cells that are not utilized at the end user. Therefore, when a cell is dropped, all the following cells from the same frame are discarded. Following frames that depend on the lost frame will be discarded as well. By applying this modification, the buffer is cleared from cells that are not used at the end user allowing more buffer space to be used by other cells, and therefore, decreasing the number of overflow incidents at the buffer.

4.2 *Overview*

4.2.1 Network Model and Assumptions

In this work, we apply the estimation methodology on buffer management schemes in access network nodes such as edge routers. Access nodes usually deploy many network algorithms such as call admission control, rate allocation, and traffic shaping. Therefore, it will be suitable to accompany such algorithms with an estimation algorithm to provide an evaluation of the connections quality. In addition, access nodes will have more resources to carry

out application-specific algorithms such as the presented estimation algorithm.

A single video traffic source is used in this work to study the performance of the network node as well as to present the accuracy of the estimation method for a simple case. Using a single video stream is reasonable because access nodes and edge routers typically perform per-flow queueing because of the other network functions they perform such as call admission control and rate allocation. As will be discussed in the case study in Section 4.8, admission control is one example of applications that maintain a queue per flow and can utilize an estimation algorithm like what we are proposing.

4.2.2 Goals

The estimation analysis details for the system is explained in this section. The Markov chains estimation model is aimed to include the following properties.

- **Cell level measurement:** Because of the variable size of video frames and slices, frame level or slice level loss measurement may not be accurate. Therefore, we measure video frame loss in terms of fixed-size cells, which result in a more accurate metric to the actual video loss.
- **Pessimistic system evaluation:** The order of events in the system model results in a pessimistic evaluation of the losses, which produces an upper bound on the video loss that corresponds to certain network conditions. The pessimistic evaluation will become more clear when the algorithm is discussed in Subsection 4.3.1.
- **Goodput evaluation:** The model is designed to accommodate video traffic characteristics in evaluating the goodput of simple buffer management schemes. In smart buffer management schemes, it accommodates video characteristics in dropping cells to maximize the goodput of the video stream and to reduce sending unusable cells to the destination.
- **Deterministic traffic model:** The order of frames within a GOP corresponds to the video frame order in their transmission order. This introduces source modeling

accuracy as compared to source models that use statistical arrangement of frames throughout the video stream.

- **Flexible model:** The estimation model is implemented for several buffer management schemes such as TD, PBS and TBS and their smart versions.

4.3 Queueing Analysis

4.3.1 Queueing Model Description

To transmit a video stream in the system, frames are segmented into cell sizes according to the network protocol applied, and then transmitted using the system slots. We will define the *frame inter-arrival time* (FIT) as the time between the beginning of two consecutive frames. It is equal to a fixed number of slots sufficient to transmit the maximum frame size⁴ (see Figure 4). Since video frames are variable in size, a frame does not necessarily fill all the slots in a FIT; therefore, a varying number of slots at the end of a FIT may remain empty, as shown in Figure 5(a). This produces ON-OFF periods in the transmission channel: a burst of cells (frame) is transmitted, then it is followed by unused slots till the next cell burst starts.

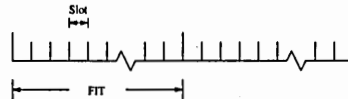


Figure 4: Slotted queueing system.

Cells that arrive at the buffer are either accepted or discarded according to the buffer management scheme applied. In simple buffer management schemes, a cell is dropped only when buffer occupancy exceeds the threshold value corresponding to the buffer management scheme applied, and no other cells are dropped even though the others cells may not be usable at the end user. Because of the information dependencies among the video frames, other frames that are dependent on the dropped frame cannot be fully reconstructed when

⁴The maximum frame size in this work is 20 slots.

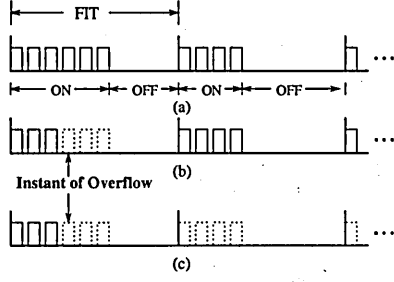


Figure 5: (a) ON-OFF source pattern. (b) Effect of buffer overflow in a B-Frame (frame drop). (c) Effect of buffer overflow in an anchor frame (GOP drop).

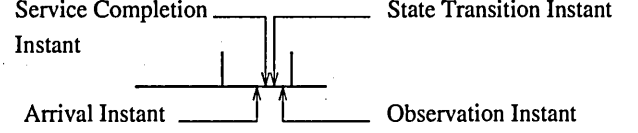


Figure 6: Pessimistic event order within a slot.

they arrive at their destination and are discarded⁵. In smart buffer management schemes, however, dropping a cell during a receipt of a frame results in dropping ensuing cells that belong to that frame and all following frames that are dependent on the dropped frame. Hence, when a B-frame causes the buffer to overflow, only that B-frame is discarded, as shown in Figure 5(b). However, when an anchor frame causes the buffer to discard cells, other frames in the GOP are also discarded as shown in Figure 5(c). In our system, we will not empty the buffer of the arrived slots that belong to a discarded frame before the overflow happens. This is an approximation to the goodput to accommodate the definition of TD, PBS, and TBS, and to simplify the theoretical analysis significantly. Emptying the buffer will also make it difficult to implement in practice. Buffered traffic is then served in a FIFO fashion to be transmitted over the output link to its destination.

4.3.2 Markov Chain Definition

To analyze the system described in the previous section, a Markov chain was embedded at the observation instants at the end of each slot. In this study, we assume the order of simulation events is as shown in Figure 6. An arrival event occurs at the start of a cell arrival at the beginning of the slot and a departure event occurs when a cell transmission is completed and after the arrival event occurs. Therefore, if the buffer is full and both a departure and an arrival occur in a given slot, the queue overflows, producing a pessimistic evaluation of

⁵Error concealment algorithms can be applied to corrupted frames and some information can be retrieved; however, their effect is not considered here in the goodput analysis.

the system's performance (i.e., maximal loss is incurred). Statistics are gathered at the end of the slot and after the departure completion. Notice that the GOP structure shown in Figure 2 is used in our analysis.

The queueing system with different queue management schemes will result in slightly different models for the system. We will first describe the characteristics of the system in general then we present the specific details for each dropping scheme, which extends our work presented in [4, 3]. The queueing system is analyzed using a five-dimensional Markov chain with a state representation (t, w, z, y, x) , where:

- t is the slot number within a FIT, such that $t \in \{0, 1, 2, \dots, T-1\}$, which is a function of line speed and cell size in the studied system;
- w is the frame number in a GOP, such that $w \in \{0, 1, 2, \dots, W-1\}$ ⁶;
- x indicates if the current frame has completely arrived or not, such that $x \in \{0, 1\}$ (see Subsection 4.3.3);
- z indicates the state of the buffer and whether or not the incoming cells are usable. The possible state values for z depend on the applied buffer management scheme; hence, it varies between 4 to 7 different states in this work. These states are mainly used to calculate the goodput of the system; and
- y is the instantaneous length of the queue, such that $y \in \{0, 1, \dots, B\}$.

Given the state description above, we notice that the repeating structure of the GOP may be exploited to realize a p -cyclic Markov chain [54] representing the transition probability between slots of a GOP. The state machine's probability transition matrix of the estimation model has the block form given in equation (4). \mathbf{P} is a $TW \times TW$ p -cyclic block matrix that contains a top level description of the cyclic state machine mentioned earlier, while the total size of \mathbf{P} is $O(WTB)$ for all of the buffer management schemes. It defines

⁶The frames in a GOP are numbered according to their transmission order, therefore, the B-frames numbered 1 and 2 correspond to the last two frames of the previous GOP.

$$\mathbf{P} = \begin{matrix} \tau_{\downarrow}^{\rightarrow} \\ 0 \\ 1 \\ \vdots \\ TW-1 \end{matrix} \begin{pmatrix} 0 & 1 & \dots & TW-1 \\ 0 & \mathbf{R}(0,0) & 0 & \dots & 0 \\ 0 & 0 & \mathbf{R}(1,0) & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & 0 \\ 0 & 0 & \dots & \mathbf{R}(T-2, W-1) & \dots & 0 \\ \mathbf{R}(T-1, W-1) & 0 & \dots & 0 & \dots & 0 \end{pmatrix} \quad (4)$$

the t and w dimensions of the Markov chain. Each submatrix represents a transition probability matrix between consecutive slots in the GOP. The dimension of the block matrix is equal to the total number of slots in the GOP whether they are occupied with cells or empty. We define the value τ , where $\tau = \tau(t, w) = wT + t$, to reference slots in the GOP when distinction between frame boundaries is not necessary.

In the remainder of this section and in Section 4.4, we present the analysis for the PBS scheme as an example of how the Markov chain is implemented. In Section 4.5, the difference in queueing analysis between PBS and PBS-D is discussed. We discuss the differences between PBS and other schemes in Section 4.6.

4.3.3 $\mathbf{R}(t, w)$ Matrices Description

The submatrix $\mathbf{R}(t, w)$ describes the system behavior during the transition between the end of slot τ and the end of slot $\tau \oplus 1$ ⁷. Cells arrive in bursts (frames) according to the video frame structure. Cell bursts are usually followed by unused slots; therefore, a slot can either be part of the burst or part of the unused period. We notice that in any slot, exactly one of two cases can occur: a cell or no cell arrival. We describe each case with a separate submatrix in the Markov chain transition matrix because of their different effects on the system. These two matrices are used to define the x dimension in the Markov chain. The first matrix, denoted as $\mathbf{C}_a = \mathbf{C}(t, w, 1)$, represents the case of the system transitioning from a slot τ to slot $\tau \oplus 1$, given that an arrival occurs during slot $\tau \oplus 1$. Similarly, the second matrix, denoted as $\mathbf{C}_n = \mathbf{C}(t, w, 0)$, represents the case of the system transitioning from a slot τ to slot $\tau \oplus 1$, given that *no* arrival occurs during slot $\tau \oplus 1$. Using these two

⁷Note that the symbol \oplus denotes modulo- (TW) addition.

matrices, we can write the general structure of $\mathbf{R}(t, w)$, for $t \in \{1, 2, \dots, T - 2\}$, in the following form:

$$\mathbf{R}(t, w) = \begin{matrix} & \begin{matrix} x_{\downarrow}^{\rightarrow} & 1 & 0 \end{matrix} \\ \begin{matrix} 1 \\ 0 \end{matrix} & \begin{pmatrix} \mathbf{C}_a & \mathbf{C}_n \\ 0 & \mathbf{C}_n \end{pmatrix} \end{matrix} \quad (5)$$

The submatrix in equation (5) defines the predominant structure of the slot transition matrix. The first row in $\mathbf{R}(t, w)$ represents a transition vector from the “busy slot” state to itself (first column) or to the “idle slot” state (second column). The second row represents a transition vector from the “idle slot” state to itself (second column) or to the “busy slot” state (first column). Once the system is in the “idle slot” state, it can only make a transition to an “idle slot” state, except during the last slot in a frame where it always transitions to a “busy slot” state. Therefore, the system transitions to/from the first slot of a frame have special structures.

The transition matrix *from* the first slot in a frame, $\mathbf{R}(0, w)$, has the matrix block form given in (6). Notice that equation (6) represents a system transition between the end of the first slot in a frame and the end of the second slot. Since there is always an arrival at the first slot of a frame⁸, only a “busy slot” state is needed at that slot to describe the system. Therefore, $\mathbf{R}(0, w)$ reduces to one vector.

$$\mathbf{R}(0, w) = \begin{matrix} & \begin{matrix} x_{\downarrow}^{\rightarrow} & 1 & 0 \end{matrix} \\ 1 & \begin{pmatrix} \mathbf{C}_a & \mathbf{C}_n \end{pmatrix} \end{matrix} \quad (6)$$

$$\mathbf{R}(T - 1, w) = \begin{matrix} & \begin{matrix} x_{\downarrow}^{\rightarrow} & 1 \end{matrix} \\ \begin{matrix} 1 \\ 0 \end{matrix} & \begin{pmatrix} \mathbf{C}_a \\ \mathbf{C}_a \end{pmatrix} \end{matrix} \quad (7)$$

Similarly, $\mathbf{R}(T - 1, w)$ in equation (7) represents the system transition between the end of the last slot in the frame and the end of the first slot of the following frame, $\mathbf{R}(T - 1, w)$. Since there is always an arrival in the first slot of a frame, there is no transition to an “idle

⁸We assume that a frame has a minimum size of one cell.

slot” state from the last slot of the previous frame. These states are also used to calculate the goodput for the PBS scheme.

4.3.4 $C(t, w, x)$ Structure

Consider the previously mentioned $C(t, w, x)$ submatrices. In each submatrix, we can define four different states (z dimension) to describe the buffer’s behavior; we will denote them as *accept*, *drop frame*, *drop GOP*, and *drop GOP**.

- In the *accept* state, the buffer performs the normal operation of receiving and enqueueing cells, if adequate space exists.
- When the buffer overflows, its state will change depending on the cell type that is discarded. If the buffer overflows while receiving a B-frame, the buffer state transitions to the *drop frame* state because the cells of that frame are the only cells to be counted as unusable before it returns to the *accept* state.
- If the buffer overflows during receiving an anchor frame, however, the buffer transitions to the *drop GOP* state because cells of the rest of the frames in the current GOP are counted as unusable in the goodput estimation.
- In the GOP transmission order, frames are rearranged according to interdependencies among frames. This rearrangement introduces an overlap in the boundaries between successive GOPs because the last two B-frames in a GOP are dependent on the I-frame of the next GOP, and that I-frame has to be transmitted before these B-frames. During the *drop GOP** state, the I-frame of the next GOP is considered usable and the following two B-frames are considered unusable. In the *drop GOP* state, the frames are considered unusable until buffer state transitions to *drop GOP** when it starts receiving a new I-frame. The difference between *drop GOP* and *drop GOP** is that, in the former, the I-frame that belongs to the current GOP is considered unusable while in the latter, the I-frame that belongs to the next GOP is considered usable.

Each of the $C(t, w, x)$ matrices has the general block matrix structure given in equation (8) (where *Drop* is abbreviated as D'). Before describing the specific structure of $C(t, w, x)$,

we will elaborate on the possible forms of $\mathbf{D}_{ij}(t, w, x, z)$; by doing so, the specific structure of $\mathbf{C}(t, w, x)$ will become more clear.

$$\mathbf{C} = \begin{matrix} \begin{matrix} z \downarrow \\ \text{accept} \\ D' \text{ frame} \\ D' \text{ GOP} \\ D' \text{ GOP*} \end{matrix} & \begin{matrix} \text{accept} & D' \text{ GOP} & D' \text{ frame} & D' \text{ GOP*} \end{matrix} \end{matrix} \begin{pmatrix} \mathbf{D}_{11} & \mathbf{D}_{12} & \mathbf{D}_{13} & \mathbf{D}_{14} \\ \mathbf{D}_{21} & \mathbf{D}_{22} & \mathbf{D}_{23} & \mathbf{D}_{24} \\ \mathbf{D}_{31} & \mathbf{D}_{32} & \mathbf{D}_{33} & \mathbf{D}_{34} \\ \mathbf{D}_{41} & \mathbf{D}_{42} & \mathbf{D}_{43} & \mathbf{D}_{44} \end{pmatrix} \quad (8)$$

4.3.5 $\mathbf{D}_{ij}(t, w, x, z)$ Matrix Description

The matrix $\mathbf{D}_{ij}(t, w, x, z)$ is a $(B + 1) \times (B + 1)$ matrix; it describes the instantaneous buffer length at each observation instant given the submatrix position in \mathbf{P} , which defines the y dimension in the Markov chain. We use four types of $\mathbf{D}_{ij}(t, w, x, z)$ matrices in our estimation model to define $\mathbf{C}(t, w, x)$ matrices; these matrices are: $\mathbf{D}_{f_l} = \mathbf{D}(t, w, 1, 0)$, $\mathbf{D}_e = \mathbf{D}(t, w, x, 1)$, $\mathbf{D}_{t_l} = \mathbf{D}(t, w, 1, 2)$, and $\mathbf{D}_{c_l} = \mathbf{D}(t, w, 1, 3)$, where l is the buffer threshold value corresponding to the current frame type. In PBS for example, $l = B$ for anchor frames and $l = T$ for B-frames. The matrix \mathbf{D}_{f_l} represents the buffer operation under normal conditions when there are cell arrivals and no buffer overflow, and all cells are considered usable. \mathbf{D}_{f_l} has the form given in equation (9), where α (in cells/slot) denotes the buffer's service rate⁹ and the number of non-zero rows equals to the threshold value, l . Since there is a cell arrival, the buffer size either increases (when there are no cell departures during a given slot) or remains constant (when another cell departs from the buffer) depending on the occurrence of a cell departure.

$$\mathbf{D}_{f_l} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & \alpha & 1 - \alpha & & \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ & & 0 & \alpha & 1 - \alpha \\ 0 & \dots & & 0 & 0 & 0 \\ 0 & \dots & & 0 & 0 & 0 \end{bmatrix} \quad (9)$$

The matrix \mathbf{D}_e describes the buffer operation when it is not accepting cells because there are no cell arrivals at all. Therefore, \mathbf{D}_e has the form given in equation (10). Since

⁹The system is analyzed with a probabilistic (statistical) buffer service rate α .

the buffer is not accepting new cells, the buffer size either remains constant or decreases, depending on the occurrence of a cell departure.

The matrix D_{t_l} describes the buffer operation at the instant of buffer overflow when it starts counting arriving cells as unusable. D_{t_l} has the form given in equation (11), where the number of zero rows equals to the threshold value. Since the buffer cannot exceed the corresponding threshold value at the point of overflow, the buffer size either remains constant (when there are no cell departures from the buffer during that slot) or it decreases (when there is a cell departure from the buffer).

Finally, the matrix D_{c_l} describes the buffer operation when processing unusable cells. It has the form given in equation (12)¹⁰. The matrices D_{f_l} , D_e , D_{t_l} , and D_{c_l} are used to specify the the entries of C for each slot. In summary, the slot number is defined using P and the arrival process for each slot is defined using R . The system behavior depends on the occurrence of cell arrivals. If there is a cell arrival ($C = C_a$) in the current slot, the possibility of buffer overflow will be indicated by the buffer occupancy that is preserved in D_{ij} . If an overflow occurs, the system's next state and the time it spends in that state are governed by type of the current frame, which is indicated by the current slot number. The system transitions back to *accept* state after frame dependency between frames ends. The estimation algorithm is mainly implemented by the definition of C for each slot because it contains the state space the governs cell loss.

$$D_e = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ \alpha & 1-\alpha & 0 & & \\ 0 & \alpha & 1-\alpha & & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \alpha & 1-\alpha \end{bmatrix} \quad (10)$$

$$D_{t_l} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & \alpha & 1-\alpha & 0 \\ 0 & \cdots & 0 & \alpha & 1-\alpha \end{bmatrix} \quad (11)$$

¹⁰Notice that $D_{c_l} = D_{f_l} + D_{t_l}, \forall l$

$$\mathbf{D}_{c_l} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & \alpha & 1-\alpha & & \\ \vdots & 0 & \alpha & 1-\alpha & \vdots \\ & \ddots & \ddots & \ddots & 0 \\ & & & \alpha & 1-\alpha & 0 \\ 0 & \dots & & & \alpha & 1-\alpha \end{bmatrix} \quad (12)$$

4.4 Details of Matrix Description

4.4.1 \mathbf{C}_a Matrix Description

Recall that, in Subsection 4.3.3, the \mathbf{C}_a matrix was defined as a block matrix which contains the transition probabilities of the system progressing from slot τ to slot $\tau \oplus 1$, given that an arrival occurs during slot $\tau \oplus 1$. Consider the frame, w , to which an arriving cell belongs. The behavior of the buffer will vary significantly based on the frame type being transmitted. Therefore, several formulations for \mathbf{C}_a may be derived based on the frame type and slot number under consideration.

It should be noted that all forms of the matrices described in this subsection have the same dimensions; some formulations, however, can be reduced because some states are superfluous.

\mathbf{C}_a for I-frame

The transition matrix between the slots in the I-frame, $\mathbf{C}_a = \mathbf{C}(t, 0, 1)$, where $t \in \{0, 1, \dots, T-2\}$, have the form given in equation (13). Notice that buffer overflow in the *accept* and *drop GOP** states causes the system to transition to the *drop GOP* state (in the first and fourth rows) rather than to the *drop frame* state. Once the buffer is in the *drop GOP* state, the system remains in it until the end of the GOP.

$$\mathbf{C}(t, 0, 1) = \begin{bmatrix} \mathbf{D}_{f_B} & \mathbf{D}_{t_B} & 0 & 0 \\ 0 & \mathbf{D}_{c_B} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \mathbf{D}_{t_B} & 0 & \mathbf{D}_{f_B} \end{bmatrix} \quad (13)$$

The transition matrix from the last slot, $\mathbf{C}_a = \mathbf{C}(T-1, 0, 1)$, has a different structure than other slots in this frame, as it represents the transition at the boundaries of the frame.

The transition from the last slot in the I-Frame has the form given in equation (14). An overflow at the transition between an I-frame and a B-frame implies that a cell discard occurs at the first slot of the B-frame. Therefore, a buffer overflow at this point causes a transition to the *drop frame* state rather than a transition to a *drop GOP* state. Notice also that, if the buffer is in the state *drop GOP**, cells from the B-frame begin to be counted as unusable (as shown in the fourth row of equation (14)). The threshold value for B-frames in the D_{ij} matrices changes from B to T because of how the PBS scheme works.

$$C_a = C(T-1, 0, 1) = \begin{bmatrix} D_{f_T} & 0 & D_{t_T} & 0 \\ 0 & D_{c_T} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & D_{c_T} \end{bmatrix} \quad (14)$$

$$C(t, w, 1) = \begin{bmatrix} D_{f_B} & D_{t_B} & 0 & 0 \\ 0 & D_{c_B} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (15)$$

C_a for P-frame

The transition matrix between slots corresponding to a P-frame, $C_a = C(t, w, 1)$, where $t \in \{0, 1, \dots, T-2\}$ ¹¹, has the block matrix given in equation (15). Notice that the system can be either in the *accept* state (when the buffer has adequate space to accept cells) or in the *drop GOP* state, and it transitions to only one of these states because if a cell arrival causes the buffer to overflow, the system transitions to *drop GOP* state and remains there until a new I-frame arrives.

$$C_a = C(T-1, w, 1) = \begin{bmatrix} D_{f_T} & 0 & D_{t_T} & 0 \\ 0 & D_{c_T} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (16)$$

The transition matrix from the last slot, $t = T-1$, has the form given in equation (16). As mentioned in the I-frame case, an overflow will cause the system to transition to the *drop frame* state because the received cell belongs to the next B-frame. Notice that the

¹¹In our example shown in figure 2, P-frames occur during frames with $w \in \{3, 6, 9\}$.

transition matrices for the P-frames are similar to the transition matrices for the I-frames except that the *drop GOP** state is omitted from the case corresponding to P-frames.

C_a for B-frame

B-frames have widely varying slot matrix structures that depend on t and w . The B-frames' slot transition matrix structure, $C_a = C(t, w, 1)$, where $t \in \{0, 1, \dots, T-2\}$, except for the last two frames¹², has the block matrix form given in equation (17). Notice that a cell can be counted as unusable either because of a cell loss within the frame or because of a GOP loss. Cell discard, when the buffer occupancy exceeds T , in a B-frame will only cause a transition from *accept* state to *drop frame* state, indicated by submatrix D_{tT} in the first row in equation (17).

$$C(t, w, 1) = \begin{bmatrix} D_{fT} & 0 & D_{tT} & 0 \\ 0 & D_{cT} & 0 & 0 \\ 0 & 0 & D_{cT} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (17)$$

$$C_a = C(T-1, w, 1) = \begin{bmatrix} D_{fT} & 0 & D_{tT} & 0 \\ 0 & D_{cT} & 0 & 0 \\ D_{fT} & 0 & D_{tT} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (18)$$

The last slot in a B-frame, $t = T-1$, has one of three different structures depending on the next frame's type. If a B-frame is followed by another B-frame, the last slot's transition matrix is described in equation (18).

Notice that, if the system is in the *drop frame* state, it transitions to the *accept* state to start accepting the following B-frame as usable. If the system is in *drop GOP* state, however, it remains at the same state to continue discarding the cells of the next frame as unusable. The same scenario occurs if the B-frame is followed by a P-frame, as given in equation (19).

Finally, if the B-frame is followed by an I-frame, the matrix for the last slot will have the form given in equation (20). A transition from *drop GOP* to *drop GOP** occurs in equation

¹²These B-frames correspond to frames with $w \in \{4, 5, 7, 8, 10, 11\}$.

(20) indicating that new I-frame cells should not be affected by the current GOP condition and are accepted as usable if the buffer does not overflow. However, it still conveys the current GOP condition to the following two B-frames to which they belong.

For the slots in the last two B-frames in the GOP¹³, the C_a matrix has the form given in equation (21). Since these two frames overlap with the I-frame of the next GOP, another state is added to handle this overlap. As mentioned before, the *drop GOP** is added to indicate the cause of dropping the GOP. Notice that, in either case, these two frames are counted as unusable as they are dependent on the preceding P-frame in the GOP and on the I-frame of the next GOP.

$$C(T-1, w, 1) = \begin{bmatrix} D_{f_B} & D_{t_B} & 0 & 0 \\ 0 & D_{c_B} & 0 & 0 \\ D_{f_B} & D_{t_B} & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (19)$$

$$C_a = C(T-1, w, 1) = \begin{bmatrix} D_{f_B} & D_{t_B} & 0 & 0 \\ 0 & D_{t_B} & 0 & D_{f_B} \\ D_{f_B} & D_{t_B} & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (20)$$

$$C(t, w, 1) = \begin{bmatrix} D_{f_T} & 0 & D_{t_T} & 0 \\ 0 & D_{c_T} & 0 & 0 \\ 0 & 0 & D_{c_T} & 0 \\ 0 & 0 & 0 & D_{c_T} \end{bmatrix} \quad (21)$$

$$C_a = C(T-1, 1, 1) = \begin{bmatrix} D_{f_T} & 0 & D_{t_T} & 0 \\ 0 & D_{c_T} & 0 & 0 \\ D_{f_T} & 0 & D_{t_T} & 0 \\ 0 & 0 & 0 & D_{c_T} \end{bmatrix} \quad (22)$$

The transition matrix across the boundary between the two B-frames, $t = T-1$ and $w = 1$ in our example, i.e. from the last slot of one B-frame to the first slot of the immediately succeeding B-frame, has the matrix form given in equation (22). Notice that it is similar to equation (18) but with the added transition to/from the *drop GOP** state.

¹³The last two B-frames of a GOP correspond to frames with $w \in \{1, 2\}$ in the following GOP.

Finally, the transition between the last slot in the last B-frame in a GOP and a succeeding P-frame has the matrix form given in equation (23). Here, the cell discard because of the current GOP loss is terminated and the buffer returns to the *accept* state unless a new buffer overflow occurs while accepting the first slot in the P-frame. Notice that cells continue to be counted as unusable if the system is in *drop GOP* state, which implies a buffer overflow while receiving the previous I-frame.

$$\mathbf{C}_a = \mathbf{C}(T-1, 2, 1) = \begin{bmatrix} \mathbf{D}_{f_B} & \mathbf{D}_{t_B} & 0 & 0 \\ 0 & \mathbf{D}_{c_B} & 0 & 0 \\ \mathbf{D}_{f_B} & \mathbf{D}_{t_B} & 0 & 0 \\ \mathbf{D}_{f_B} & \mathbf{D}_{t_B} & 0 & 0 \end{bmatrix} \quad (23)$$

4.4.2 \mathbf{C}_n Matrix Description

In the previous subsection, we presented the different structures for \mathbf{C}_a that describe the system transition to a slot, given that there is a cell arrival. Recall that, in equation (5) we defined \mathbf{C}_n to describe the system transition to a slot, given that there is no arrival.

The matrices in the case of no arrival are very similar to the case of cell arrivals; however, some modifications are required to compensate for the absence of arrivals. The matrices in the previous subsection are repeated in this subsection with the necessary modifications.

The matrix corresponding to transitions between idle slots in the I-frame, $\mathbf{C}_n = \mathbf{C}(t, 0, 0)$, where $t \in \{0, 1, \dots, T-2\}$, now has the structure given in equation (24). Since there are no arrivals, there are neither buffer overflow nor acceptance. Therefore, the transition matrix is fully described using \mathbf{D}_e . Notice that all the non-zero elements appear on the matrix diagonal, which means that the system stays at the same state until the first cell of the next frame arrives.

$$\mathbf{C}_n = \mathbf{C}(t, 0, 0) = \begin{bmatrix} \mathbf{D}_e & 0 & 0 & 0 \\ 0 & \mathbf{D}_e & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{D}_e \end{bmatrix} \quad (24)$$

$$\mathbf{C}_n = (t, w, 0) = \begin{bmatrix} \mathbf{D}_e & 0 & 0 & 0 \\ 0 & \mathbf{D}_e & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (25)$$

The transition between idle slots in the P-frame, has the matrix structure given in equation (25). As mentioned in the previous subsection, the slots in the P-frame can be fully described using the *accept* and the *drop GOP* states.

The cells for the B-frame, except for the last two B-frames, have the form given in equation (26).

$$C_n = C(t, w, 0) = \begin{bmatrix} D_e & 0 & 0 & 0 \\ 0 & D_e & 0 & 0 \\ 0 & 0 & D_e & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (26)$$

$$C_n = C(t, w, 0) = \begin{bmatrix} D_e & 0 & 0 & 0 \\ 0 & D_e & 0 & 0 \\ 0 & 0 & D_e & 0 \\ 0 & 0 & 0 & D_e \end{bmatrix} \quad (27)$$

For the last two B-frames, we have the form given in equation (27). All the states are needed in this case to handle frame overlap with frames from the following GOP.

4.5 PBS-D Queueing Analysis

The main difference between PBS and PBS-D lies in how the unusable cells are handled. We saw that, in the PBS scheme, cells that are not usable are admitted to the buffer, and arriving cells are discarded only if the buffer occupancy exceeds the threshold value correspond to the current frame type. In PBS-D, however, unusable cells are dropped in addition to the arriving cells that cause the buffer occupancy to exceed the threshold value.

The matrices in the PBS queueing analysis can be easily modified to implement PBS-D. The only modification occurs in states that represent unusable cells, i.e. *Drop GOP* during anchor frames, and *Drop GOP*, *Drop Frame*, and *Drop GOP** during B-frames. In these states, the buffer should be discarding the incoming cells in addition to counting them as unusable. This modification can be achieved by replacing every D_{c_B} and D_{c_T} by D_e in the C_a transition matrices in equations (13)-(23).

So far, we have presented a detailed queueing analysis for PBS and PBS-D at the cell level. We used several states to measure the goodput by accounting for lost and unusable

cells in the GOP. We also described the five dimensions of the Markov chain for the deterministic sequence of frames in the GOP. In the next section, we present the queueing analysis for the other simple and smart schemes through highlighting their differences from the PBS and PBS-D schemes.

4.6 Analysis of Other Schemes

In this section, the estimation analysis that is presented for PBS and PBS-D is extended to the TD, TD-D, TBS, and TBS-D schemes. We show the changes made to the PBS(-D) analysis to present the analysis for those schemes. Involved in the interests of space, we present in the appendices more details of the analysis.

4.6.1 TBS Queueing Analysis

As described earlier, TBS scheme uses two thresholds $T_H > T_L$ in its mechanism, which will introduce extra states. This is because when $T_L > b > T_H$, the buffer state can be in two states: (i) accepting low-priority cells if no previous cell loss is encountered, and (ii) dropping low-priority cells if the buffer occupancy exceeded T_H and did not decrease below T_L yet. To compare with the PBS case, each state that represents unusable cells is replaced by two states in TBS. Therefore, we define a new set of states to represent the z dimension in the TBS scheme. These states are *accept*, *drop GOPA*, *drop GOPD*, *drop frameA*, *drop frameD*, *drop GOPA**, *drop GOPD**, and *drop frame**. We will compare these states with PBS states to show how they simply map to each other.

- The *accept* state in the TBS scheme is exactly the same as in the PBS scheme. In both cases, this state represents cells accepted to the buffer and usable at the end user.
- The *drop GOPA* and *drop GOPD* states achieve the same purpose the *drop GOP* state does in the PBS scheme. These states represent cells that are not usable because of a previous cell loss in anchor frames. *drop GOPA* represents cells that are accepted but not usable whereas *drop GOPD* represents cells that are dropped because, after a cell drop, the buffer occupancy did not decrease below T_L yet.

$$\mathbf{C}(t, w, x) = \begin{matrix} & \text{accept} & D'GOP & D'frame & D'frame* & D'GOP* \\ \begin{matrix} \text{accept} \\ D'frame \\ D'GOP \\ D'frame* \\ D'GOP* \end{matrix} & \begin{pmatrix} D_{11} & D_{12} & D_{13} & D_{14} & D_{15} \\ D_{21} & D_{22} & D_{23} & D_{24} & D_{25} \\ D_{31} & D_{32} & D_{33} & D_{34} & D_{35} \\ D_{41} & D_{42} & D_{43} & D_{44} & D_{45} \\ D_{51} & D_{52} & D_{53} & D_{54} & D_{55} \end{pmatrix} \end{matrix} \quad (28)$$

- *drop frameA* and *drop frameD* states in the TBS scheme are similar to *drop frame* in the PBS scheme. These states represent cells that are not usable because of a previous cell loss in B-frames. As in *drop GOPA/D*, *drop frameA* represents cells that are accepted but not usable where as *drop frameD* represents cells that are dropped because, after a cell drop, the buffer occupancy did not decrease below T_L yet.
- The *drop GOPA** and *drop GOPD** states in the TBS scheme are similar to the *drop GOP** in PBS in the same fashion.
- The *drop frame** state is used in TBS only in anchor frames. When a cell is dropped in TBS, the buffer occupancy must decrease below T_L before it starts accepting low-priority cells (B-frame cells). In some cases, the buffer occupancy does not decrease below T_L during the B-frame period, and anchor frame cells will start arriving. At that point, the system needs to transition from the *drop frameA/D* states to *accept* state, and the condition requiring the buffer occupancy to decrease below T_L will be lost. Therefore, we carry the dropping status of low-priority cells from one B-frame to another through anchor frames using *drop frame**.

The detailed matrix description of \mathbf{C}_a for the TBS scheme is shown in Appendix A

4.6.2 TBS-D Queueing Analysis

The queueing analysis for the TBS-D schemes is easier than the TBS scheme. Since unusable cells are dropped, there is no need for two states for the cases of GOP loss or frame loss, which reduces the number of required states. The general \mathbf{C}_a matrix for the TBS schemes is shown in equation (28), where the total number of states used is now five states. As in PBS-D, the matrices representing states of unusable cells will be replaced by \mathbf{D}_e . Notice

also that the *drop frame** state is still needed to carry the dropping information from one B-frame to another through anchor frames.

The C_a matrices for the TBS scheme can be intuitively constructed. Due to space limitations, we list them in Appendix B.

4.6.3 TD and TD-D Queueing Analysis

The modifications required to implement TD and TD-D schemes are very simple. TD and TD-D are actually special cases of PBS and PBS-D, respectively; TD and TD-D are achieved by setting the threshold value $T = B$. This change occurs only in the C_a matrices for B-frames.

We have presented the queueing analysis of all the schemes discussed in this chapter. It was shown that, in some schemes, a simple modification to the analysis of one scheme will produce the another scheme, which is a flexibility privilege that this estimation methodology has. In the next section, we present selected performance results for the simple and smart buffer management schemes.

4.7 *Simulation and Results*

4.7.1 Simulation Environment

Consider the system illustrated in Figure 7. This system is modeled using Markov chains in order to estimate the goodput performance for the buffer management schemes mentioned earlier. Segmented frames arrive at the buffer where the order of frames (in repeating GOPs) is governed by the twelve-state machine shown in Figure 8. This way the deterministic ordering between the frame types is preserved within each GOP to resemble better source modeling accuracy. Upon arrival, if the applied buffer management scheme accepts the arrived cell, it is buffered in a FIFO fashion. Otherwise, that cell is dropped. In the smart buffer management schemes, that cell is discarded along with all successive cells that contain information dependent on that frame. In the simple buffer management schemes, however, successive cells are not dropped unless if there is no space to accommodate them. Buffered cells are served at a constant rate.

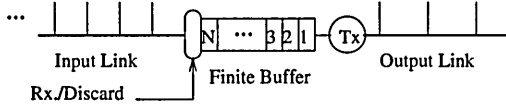


Figure 7: Queueing System Description.

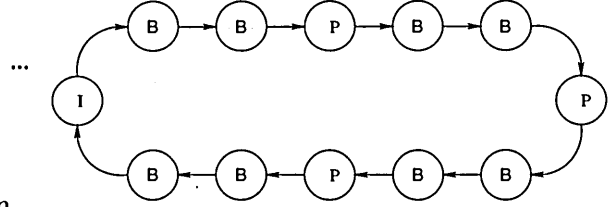


Figure 8: Markov Chain video traffic model.

4.7.2 Video Traffic Model

In this work, frames are sized according to lognormal distributions [35] that are bounded by a maximum frame size. as shown in Figure 12. The mean size of each frame type is selected according to the average of statistics for real MPEG-2 coded movies shown in [19], which were prepared by Oliver Rose of the Computer Science Institute at University of Wuerzburg. The I-, P-, and B-frames are set to have normalized mean frame sizes, $\mu_I : \mu_P : \mu_B$ of $1 : 0.3 : 0.13$ and relative standard deviations, $\sigma_I : \sigma_P : \sigma_B$ of $1 : 0.76 : 0.32$, respectively. The ratios between different frame statistics reflect the correlation between frames in MPEG-2 video streams. The inter-GOP correlation, however, is not implemented in this traffic model since the variation in the total GOP size is greatly affected by scene changes. Scenes may last for a relatively long time, which minimizes the effect of a finite buffer in improving the system performance or absorbing traffic bursts at GOP or scene level. Maintaining the previously mentioned average frame size ratios limits the system's offered load¹⁴ to 23% because, even when μ_I is maximal, $\mu_B = 0.13\mu_I$. Even though the maximum load is low, video streams are time-sensitive, which cause the input arrival rate to reach the peak rate during a cell burst (frame). This arrival pattern can cause buffer overflow when the output link is congested.

Bounded geometric frame size distributions that follow the same frame size ratios mentioned above were used to test the algorithm for several traffic models. The estimation algorithm agreement with the simulation when geometric distributions are used was similar to the lognormal distribution case. We present the results with lognormal distribution

¹⁴Offered load = (full slots/total slots) at the input link.

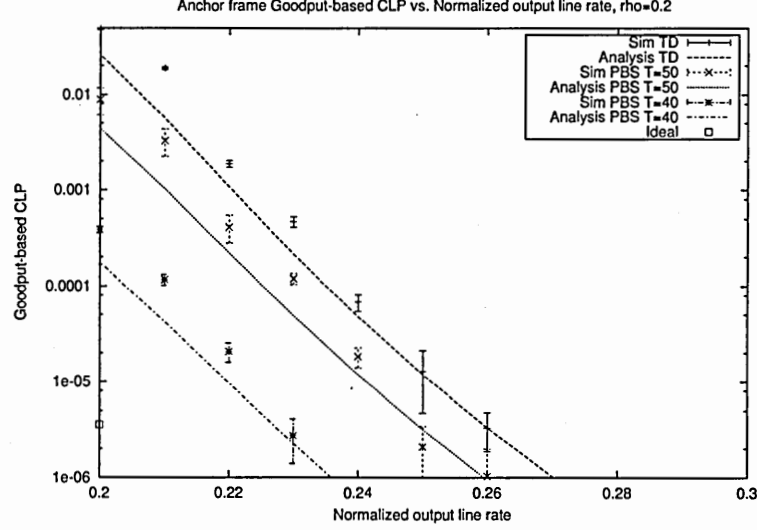


Figure 9: PBS goodput-based CLP in anchor frames.

because it is more appropriate traffic model for MPEG-2 video sequences.

The maximum load that we could reach in this system obviously shows how underutilized the input channel is when a single video stream is transmitted with peak cell rate. This highlights the need to aggregate multiple VBR video flows in the same channel to increase the bandwidth utilization, which we will investigate in future work.

A transition matrix is built for each buffer management scheme and then used to estimate the frame goodput of the system. Individual steady state probability vectors for each slot were produced using LU decomposition techniques. The analysis is compared to a discrete event simulation of the system. As previously mentioned, goodput is defined as the ratio of the cells belonging to correctly displayable frames to the total number of cells. Furthermore, we define goodput-based cell loss probability (CLP) as the loss in goodput (i.e., $goodput\text{-based } CLP = 1 - goodput$); we choose these definitions of goodput and goodput-based CLP because they are more representative measures of the integrity of the end user's video stream quality than other measures [61, 20].

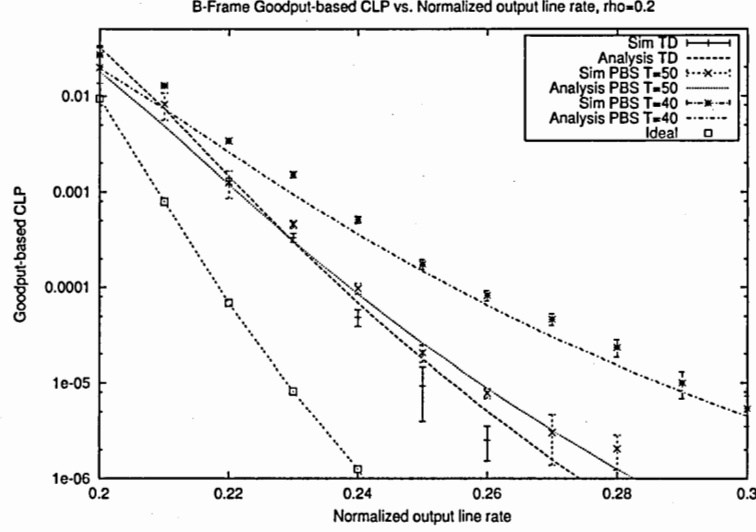


Figure 10: PBS goodput-based CLP in B-frames.

4.7.3 Buffer Management Schemes

The goodput performance of the buffer management schemes is evaluated using the results from our estimation analysis model and their corresponding simulation results. In Figures 9 – 21, we plot goodput-based CLP versus normalized output line rate¹⁵ for a buffer size of 60 cells and a load of 20%. The estimation and simulation results closely agree. In addition to the results for the buffer management schemes, a curve that represents an ideal dropping scenario is plotted to quantify the difference in performance for these schemes from the ideal scenario. The ideal scenario utilizes global information about the frame sizes and the buffer occupancy to dynamically change the threshold in the buffer. The threshold is adjusted such that the buffer accepts B-frames as long as anchor frames are not affected. P-frames are dropped if there are no B-frames that can be dropped instead. Because of the low system load and the limited simulation time, it is noticed that the simulation results approach the numerical resolution for data in the range of 10^{-6} .

The data in Figures 9 and 10 show the loss because of dropping an anchor frame and a B-frame for the PBS case, respectively. We notice that the threshold value significantly

¹⁵The normalized output rate is the ratio between the output line rate and the input line rate. It can be looked at as a measure of link congestion.

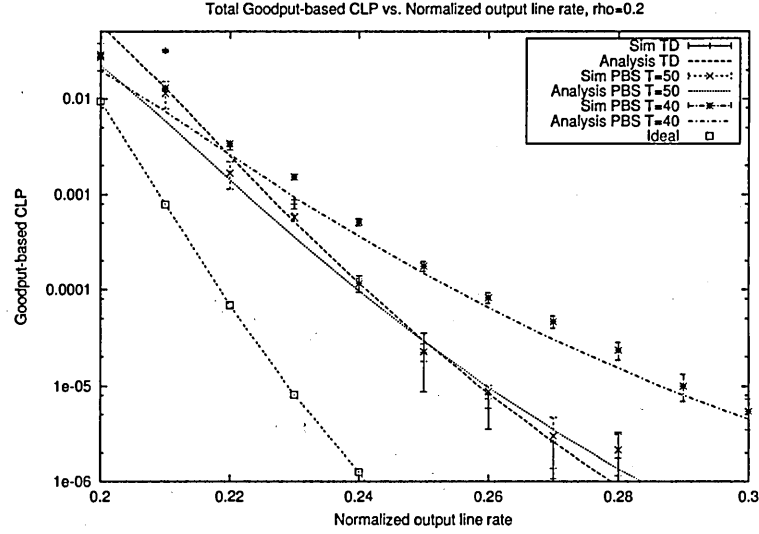


Figure 11: PBS total goodput-based CLP.

affects the loss. As the threshold value decreases, the loss in anchor frames decreases while it increases in B-frames. In PBS, anchor frames are protected because the introduced threshold limits the buffer space that B-frames can occupy while it grants anchor frames access to the entire buffer space. The threshold value has to be carefully selected because a low threshold value can cause B-frames to starve for buffer space; consequently, it can dramatically increase loss in B-frames that leads to having the total loss to be higher in PBS than TD as shown in Figure 11. An increase in the total loss can cause a degradation in the video quality due to lowering the frame rate. It is noticed that the total goodput loss in TD exceeds PBS and TBS when the normalized output rate is low. This is because individual and distributed cell loss occur more frequently in TD scheme, which results into more unusable frames in the video stream. Similar results are shown for the TBS case in Figures 13–15. The TD, PBS, and TBS schemes were evaluated in [17] for MPEG video traffic over ATM networks. The raw cell loss probability (raw CLP) was used to evaluate their performance. The behavior of these schemes was similar to the results shown in this work, however, we used the goodput-based CLP to evaluate the performance instead of raw CLP.

The effect of the threshold value in concurrently protecting anchor frames and increasing

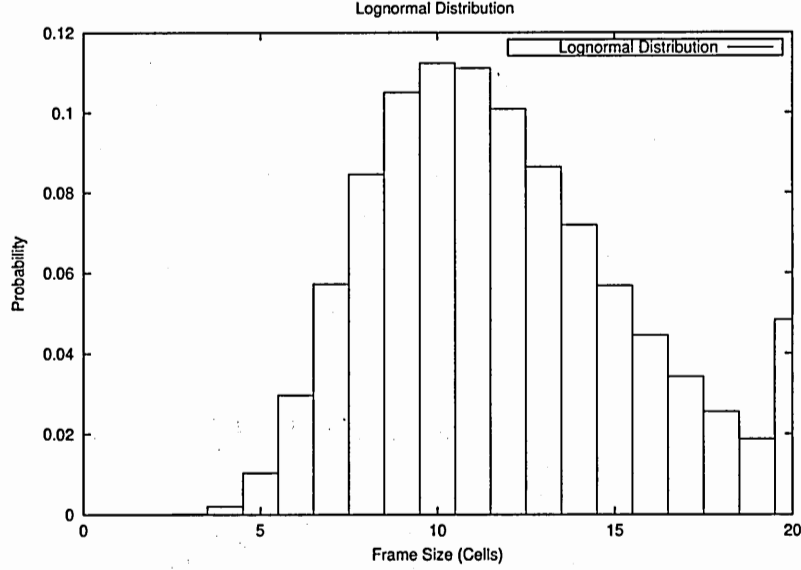


Figure 12: Bounded lognormal frame size distribution for video source models.

loss in B-frames is also obvious in the smart schemes, as shown in Figures 16–21. Notice that the estimation algorithm results closely agree with the simulation results for various buffer management schemes.

While PBS(-D) and TBS(-D) perform better than TD(-D) in protecting anchor frames, they perform worse for B-frames and result in increasing the total loss probability. The goodput-based CLP in anchor frames can be reduced without necessarily increase the goodput-based CLP in B-frames when the output bandwidth is efficiently utilized and the dropped frames are carefully selected. The curve of the ideal case suggests that there is a room for improvement in the buffer management schemes. One method to improve selective discard in threshold-based schemes is to obtain knowledge about future high-priority frames and compute the effect of accepting the current frame on the acceptance of the future high-priority frames. Hence, the node is given a set of frames (in the future) to consider before making the dropping decision. This will be equivalent to a dynamic threshold-based buffer management scheme. This method is virtually similar to POB scheme in the sense that both methods consider later high-priority frames (whether they are in buffer, as in POB, or in the future) in making the drop decision, as will be discussed in Chapter 5.

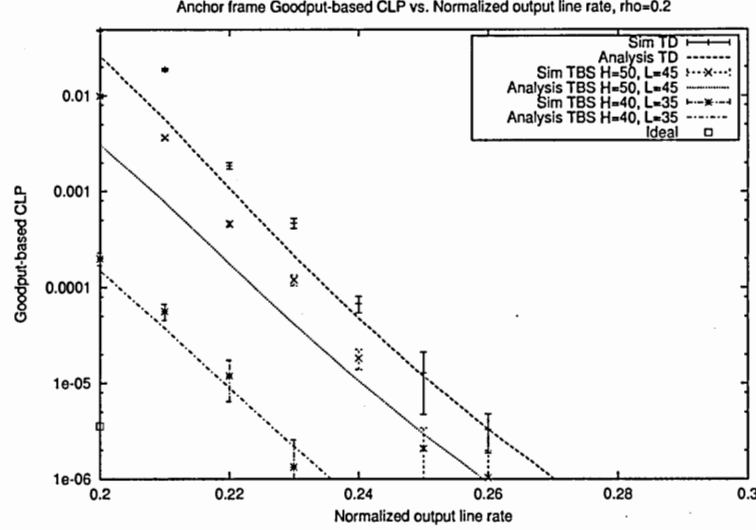


Figure 13: TBS goodput-based CLP in anchor frames.

4.7.4 Effect of Changing Load and Buffer Size

After we studied the goodput performance of the buffer management schemes, we now study the effect of the changing the buffer size and system load on the estimation accuracy and the goodput analysis. By doing this study, the sensitivity of both the system and its estimation to these factors are evaluated. To verify the estimation results, we compare the results of our analysis with the corresponding simulation results. In Figures 22–23, we plot goodput-based CLP versus normalized output line rate for different buffer sizes and different loads for the TD-D scheme.

The data in Figure 22(a) shows the goodput-based CLP plotted for several buffer sizes that range from two to four times the maximum frame size; simulation and analysis results compare favorably for that range of buffer sizes. Notice that, for an applied load of 0.2 and a buffer size of 60 cells, a goodput-based CLP of 10^{-6} can only be achieved when the normalized output rate is 0.28, which is equal to 71% bandwidth utilization instead of 20% at the input line¹⁶. This highlights the effect of buffer smoothing on bandwidth utilization and peak cell rate. Similar results are shown in Figure 22(b) for a load of 23%.

¹⁶The relationship between load and output line rate is a function of the MPEG stream burstiness. Studying this relationship is outside the scope of this work.

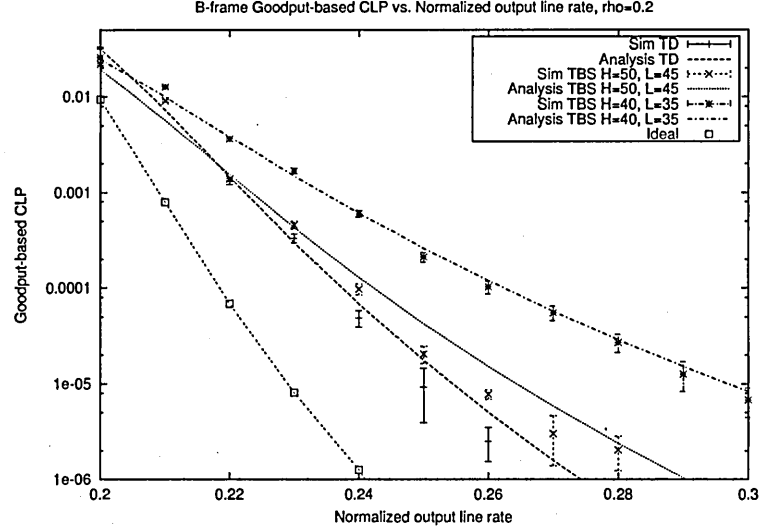


Figure 14: TBS goodput-based CLP in B-frames.

It is noticed, in both Figures 22(a) and 22(b), that the simulation results get closer to those of the analysis as the buffer size or the losses increase. These results are intuitive and expected. This is because there is more buffer stability when the size increases and there is more accuracy as goodput-based CLP increases.

In Figure 23, the effect of load on the goodput-based CLP estimation is illustrated. Our estimation algorithm works well for different system loads. As expected, when load is increased, goodput-based CLP increases as well; notice that the proposed estimation method works well in that given region of traffic load. As the load decreases, the error slightly increases due to a smaller number of cell arrivals (lower load) in the simulation that are taken into account in the calculation of the loss.

In this section, we have thus far evaluated the goodput estimation algorithm for simple and smart buffer management schemes. The simulation and estimation results agree favorably. It is noticed that having a fixed threshold value in the threshold-based schemes will result in protecting anchor frames; however, it also results in increasing the total goodput-based CLP mainly because of the goodput-based CLP increase in B-frames. We have also shown that the estimation algorithm results agree with the simulation results for different loads and buffer sizes.

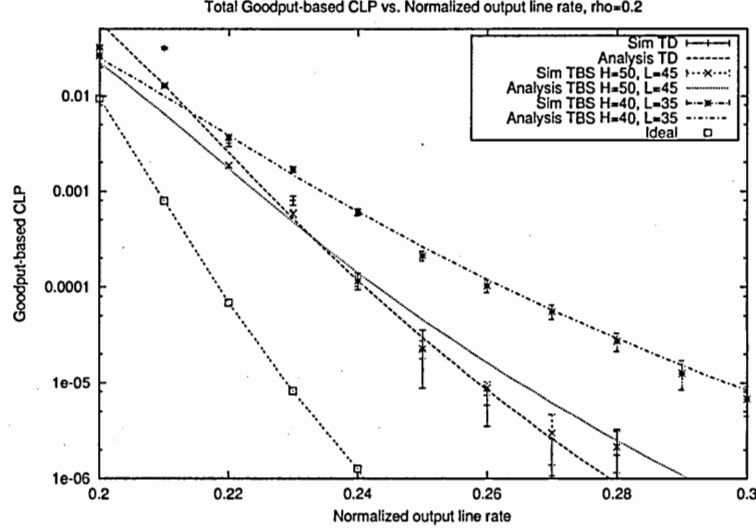


Figure 15: TBS total goodput-based CLP.

4.8 Case Study: Admission Control

As we mentioned earlier, the proposed estimation methodology can be used for several applications. In this section, we demonstrate how the estimation methodology can be used in admission control as an example. In admission control, the user requests a network connection with specific connection parameters. The user can ask for a certain bandwidth guarantee, cell loss upper limit, delay guarantees, etc. Raw CLP may not reflect the video quality loss encountered in a network connection and the goodput-based CLP needs to be used instead. We consider admission control in two cases: simple buffer management schemes and smart buffer management schemes. Through these cases, we will show how admission control can fail when goodput is not considered. We refer to this phenomenon *admission failure*. We also show that network resources can be wasted because the goodput-based algorithms at the network node are not recognized by the estimation method. We refer to this phenomenon as *bandwidth over-provisioning*.

4.8.1 Admission Control Failure

Consider a network node with a simple PBS buffer management scheme. When a user requests a connection, it provides the network node with the traffic model of the application

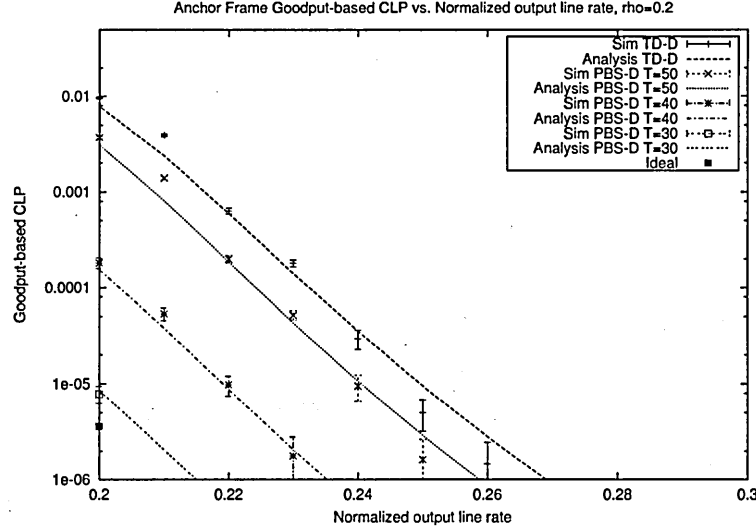


Figure 16: PBS-D goodput-based CLP in anchor frames.

that will use the connection as well as a specific connection quality. The network translates the requested quality into cell loss. To calculate the bandwidth required to provide the requested quality, an estimation methodology can be deployed. The estimation methodology can either be based on a raw cell loss ratio method, or a goodput-based method, as proposed in this work. In the raw cell loss ratio method, only the dropped cells are used to find the required bandwidth. In the goodput-based method, the dropped cells along with their dependent cells are used in finding the required bandwidth. The raw cell loss ratio method may result in smaller allocated bandwidth than the goodput-based method; however, the end user quality will be inferior to the requested quality, which results in a call admission failure. In the goodput-based estimation method, although more bandwidth may be allocated to the connection, but the end user perceived quality will be similar to the requested connection quality.

For example, consider an admission control request to a simple network node applying PBS buffer management scheme with $T = 50$. A user requests a connection for video source with a lognormal traffic model and average rate 20% of the link capacity (same as the traffic model used in the Section 4.7). The user requests a connection quality that corresponds to cell loss of 10^{-4} . In our evaluation experiments, the requested bandwidth is found to

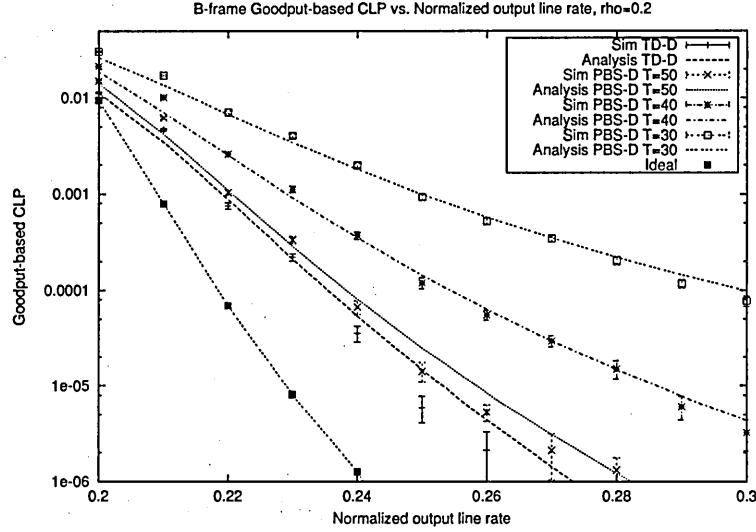


Figure 17: PBS-D goodput-based CLP in B-frames.

be 24.1% if goodput-based method is used; however, the bandwidth is found to be 23.7% when raw cell loss ratio method is used. Hence, if the raw cell loss ratio estimation method is used, the resulting video quality at the end the user will be less than what is requested, which results in call admission failure.

4.8.2 Bandwidth Over-provisioning

We consider the case of a smart node with TBS-D buffer management scheme in this subsection. As discussed in the previous subsection, a user requests a connection with a specific connection quality and the network node will allocate the appropriate bandwidth for that connection. When the goodput-based with dependency drop estimation method is used to calculate the requested bandwidth, the end user perceived quality will be similar to the requested connection quality. Now, if the goodput-based method without dependency drop is used, more bandwidth will be allocated to the connection than that of the goodput-based method with dependency drop. This is because, without dependency drop estimation, the method will assume that dropping a cell does not cause a discard of its dependent cells. Therefore, the goodput-based method without dependency drop will result in more allocated bandwidth than needed.

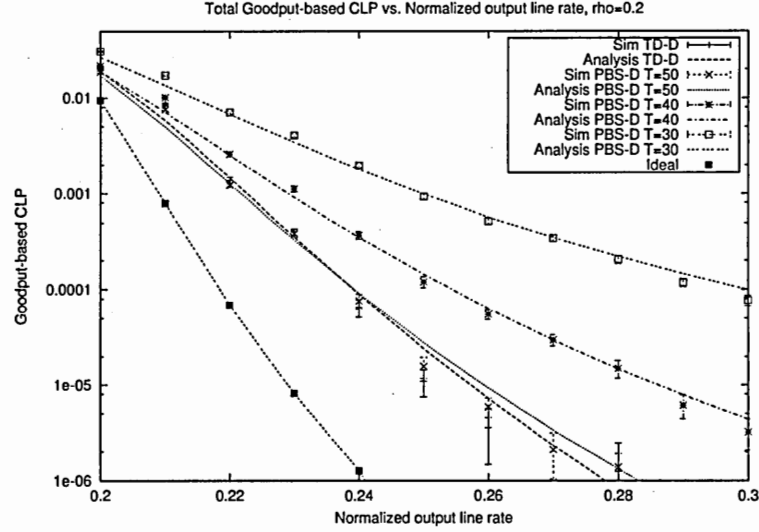


Figure 18: PBS-D total goodput-based CLP.

For example, consider an admission control request to a smart node applying TBS-D buffer management scheme with $T_H = 40$ and $T_L = 35$. We consider the same user request as in the previous subsection. As shown in Figure 15, the requested bandwidth will be 26.3% if goodput-based without dependency drop is used. However, the bandwidth will be 25.5% when dependency drop is accounted for, as shown in Figure 21. Therefore, goodput-based method without dependency drop results in more allocated bandwidth than needed.

In the above two cases, the difference in bandwidth allocation between the compared scenarios ranges between 2% to 4% of the video traffic load. This difference may result in noticeable video quality difference due to its coding structure. This case study is only an example to highlight the difference in video performance evaluation when different metrics are used. When the appropriate metric is used, a more accurate estimation of video quality is achieved, which may become significant in some cases when the difference in accuracy is critical.

4.9 Issues and Discussion

There are some limitations and considerations in the research study presented in this chapter that we will discuss in this section. The following points are some of them.

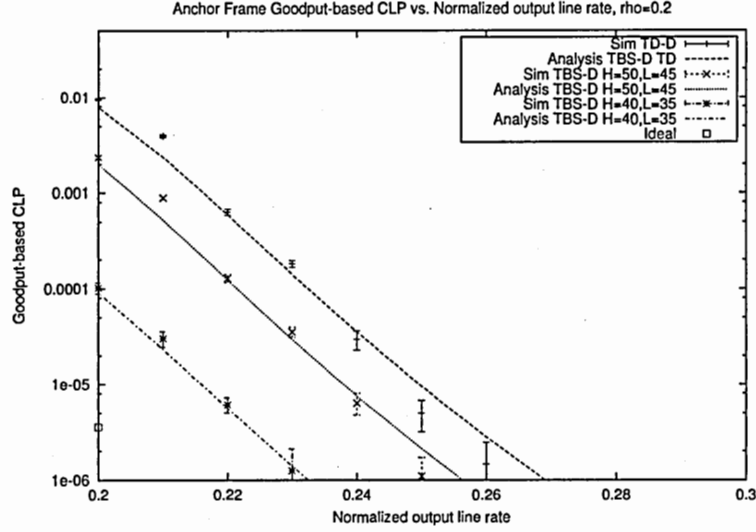


Figure 19: TBS-D goodput-based CLP in anchor frames.

- Limitations of the estimation algorithm to threshold based buffer management schemes:*

The proposed estimation algorithm is implemented for TD(-D), PBS(-D) and TBS(-D). There are other schemes that are not implemented such as push out buffer (POB). Threshold based algorithms are often used for dropping packets because they are simple and fast dropping schemes. POB and other schemes that access the buffer to drop packets are usually not desirable because of the complexity involved in accessing the buffer. Providing estimation for POB and similar schemes introduces huge complexity to the analysis and the Markov chain size. Therefore, this estimation algorithm is not implemented for such buffer management schemes.
- Goodput approximation:* There is an approximation in the goodput measurement in this study. Since the studied dropping schemes are not designed to provide transmit only complete frames, some of the transmitted frames are not complete because an overflow occurred during their reception. However, subsequent frames that are dependent on the partially accepted frame are discarded completely. Dropping schemes that are video-specific can be designed to guarantee correctness and completeness of all transmitted frames. This is a subject in the current research we are conducting.

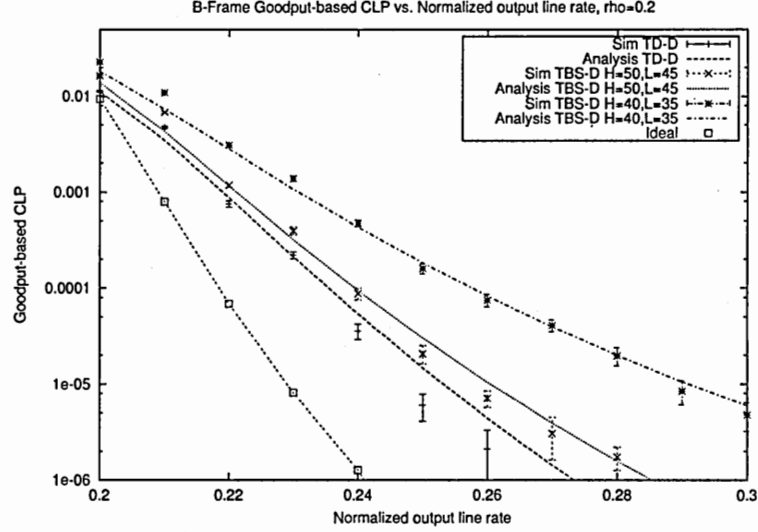


Figure 20: TBS-D goodput-based CLP in B-frames.

- *Real video*: The traffic model that is used in this work is based on statistics from real video data, however, real video traces were not used because the nature of the estimation algorithm is designed to use only traffic models. For applications that utilize changes in the traffic characteristics, the estimation algorithm can be applied by running the estimation for different traffic parameters.
- *PSNR evaluation*: PSNR is a common measure for the video quality in literature. We are considering it in the current research as one metric to evaluate the performance of new dropping schemes and existing dropping schemes. The use of statistical traffic model in the study did not facilitate using PSNR as an evaluation element either.
- *Single video source*: A single video source is used in this study to effectively study the frame goodput and verify the accuracy of the estimation algorithms. Future work will focus on generalizing the system to include multiple sources.
- *Error concealment at the node*: Error concealment algorithms do enhance the video quality by recovering from some errors caused by packet loss in the video stream. It is usually applied at the destination and not in the network. We did not pursue these error recovery algorithms because they are not designed to be implemented in

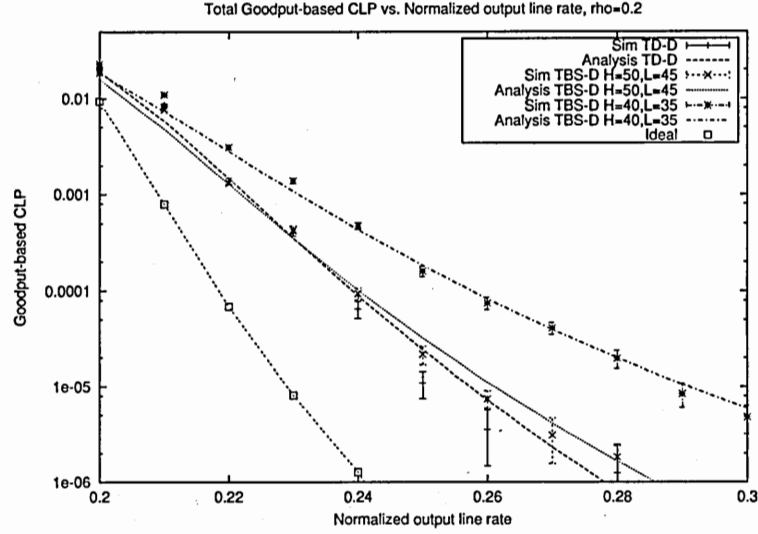
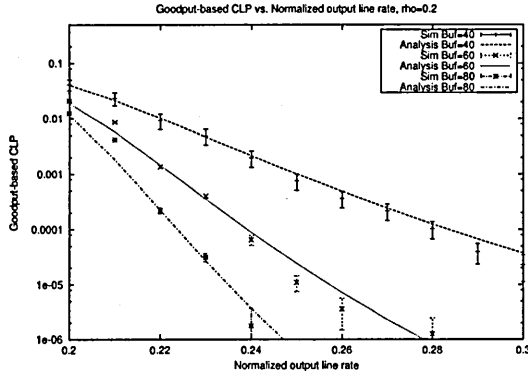
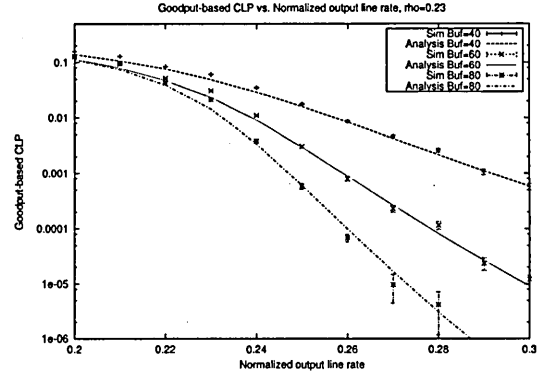


Figure 21: TBS-D total goodput-based CLP.

- a network node and they will add processing overhead to the network node.
- *Delay and Jitter*: It is important to consider buffering delay for video applications because they are time-sensitive. Usually, maximum delay is governed by the node buffer size. The maximum buffer size we used in this work is four frames that results in a maximum delay of 133ms, which is acceptable for video applications. The jitter factor often affects the transport performance of network traffic in multi-hop system. Jitter will be considered in future research about multi-hop systems.
 - *Deployment*: In case the network node is not to be altered, our estimation algorithm can be applied at the source. The source will run the estimation algorithm to determine adequate the rate to sustain the upper goodput loss limit. The selected rate is included in connection request. The network node then determines if it can establish the connection using that rate.



(a) "Low" Utilization.



(b) "High" Utilization.

Figure 22: Effect of buffer size on goodput-based CLP.

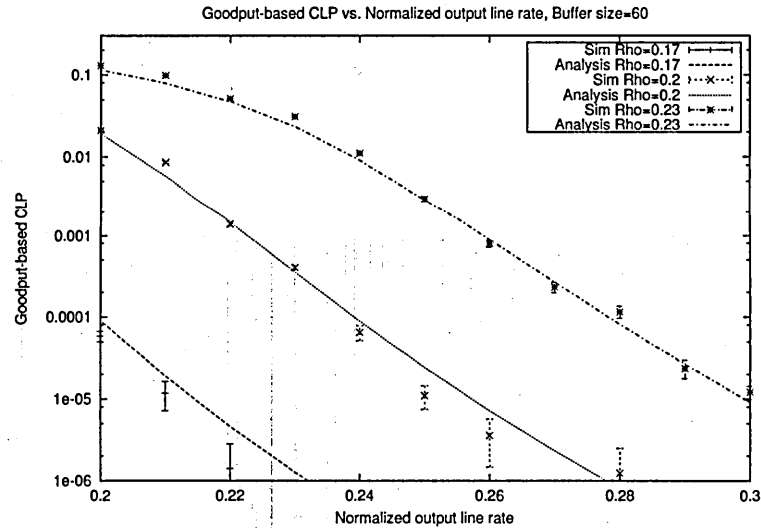


Figure 23: Effect of load on goodput-based CLP.

CHAPTER 5

MPFD: UTILITY AWARE BUFFER MANAGEMENT

5.1 *Introduction*

Several buffer management schemes have been applied to efficiently support prioritized traffic, such as video traffic. We mentioned in the previous section two approaches: (i) threshold-based approaches that drop packets upon arrival, such as PBS and TBS, and (ii) approaches like POB where the buffer contents have to be accessed to decide if a packet should be dropped. Approaches like PBS are simple to implement and can provide buffer protection for high-priority frames, but the overall performance is low because the buffer is not fully utilized as a result of the fixed partial buffer space allocated to low-priority packets. POB, on the other hand, is efficient and fully utilizes the buffer space, but it is complex to implement because it requires searching the buffer for low-priority packets to discard.

The simplicity of threshold-based schemes makes them deployable in intermediate network nodes at the expense of their relatively low efficiency. On the contrary, the complexity of the POB scheme makes it inappropriate to be deployed in network nodes despite its efficiency and higher performance.

We propose a new dropping scheme that combines the properties of both threshold-based schemes (like PBS) and POB. Multipriority frame discard (MPFD) has features that exist in both POB and PBS schemes. It has the simplicity of the PBS scheme in dropping incoming packets upon arrival and has the efficiency of POB in minimizing the number of lost frames while keeping the video quality optimal. It also accommodates video traffic characteristics to produce better performance than other dropping schemes. MPFD is based on constructing a virtual buffer that represents the future buffer occupancy using information about future video frames. The virtual buffer occupancy is used to decide whether accepting the currently arriving video frame will cause dropping a future video

frame of a higher priority or not. MPFD is similar to POB in the way it looks into a buffer (the physical buffer in POB and the virtual buffer in MPFD) to determine which frame to drop, as shown in Figure 24. The figure shows an example of dropping a low-priority frame (marked with an x) for both MPFD and POB. The same frame is dropped in both cases, but frame dropping happens at different times. In POB, the frame is dropped when the physical buffer is full, while it is dropped earlier as it arrives to the buffer in MPFD. The frame drop occurs in MPFD because accepting it will result in dropping a high-priority frame later on, as indicated by the virtual buffer.

MPFD uses information about the future video frames to construct a virtual buffer to predict a future possibility of overflow in the physical buffer. Information about future frames is sent to the network node from the source. If the future overflow occurs while receiving a high-priority frame, incoming low-priority frames are dropped in advance to avoid that overflow. This way, the physical buffer contents need not be changed when dropping a frame is necessary, and the POB complexity of searching the buffer is avoided. The information sent by the source is conveyed in a scalable fashion that is inspired by dynamic packet state approaches such as CSFQ [55].

MPFD is realized by introducing a dynamic threshold in the buffer that determines what the current buffer occupancy should be before accepting the current frame in order to avoid dropping a future higher-priority frame (or virtual buffer overflow). The distance between the current frame and the furthest future frame is denoted as the *future look-ahead step*. The dynamic threshold and its relation with the future look-ahead step and the virtual buffer will be discussed in detail in Section 5.1. Using a threshold in the buffer to trigger a frame drop makes MPFD share the simplicity of PBS.

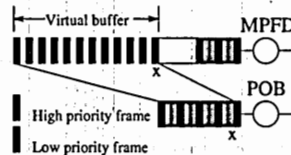


Figure 24: MPFD virtual buffer compared to POB physical buffer.

In our performance evaluations, MPFD demonstrates significant performance improvement because the loss in high-priority packets is reduced without increasing loss in low-priority frames, and the whole buffer space can be utilized by low-priority frames. In fact, the loss in low-priority packets decreases as well in some cases. We also show that MPFD can perform better than POB under certain conditions.

5.2 Overview

The MPFD design is based on the goal of achieving the performance of POB but without having its overhead complexity. In POB, a B-frame is pushed out of the buffer when an anchor frame arrives at a full buffer. In MPFD, we would like to drop that B-frame but without the complexity of searching the buffer. To avoid searching the buffer for B-frames, we would prefer dropping the B-frame when it arrives at the buffer because it involves no buffer manipulation operations. This is possible only if the network node knows that it needs to drop that B-frame to avoid dropping a future anchor frame. For the network node to know this kind of information, it needs to obtain information about future frames.

Let us assume that the network node knows the arrival pattern of anchor frames within a future look-ahead period. This information allows the node to compute the buffer occupancy during the period in which the future frames arrive. As mentioned earlier, we call the predicted buffer state (or occupancy) during that period the virtual buffer. By constructing a virtual buffer using future frames, frame information can be used to predict if a high-priority frame will overflow the buffer in the future. Given the information the network node has, if a future anchor frame is going to be dropped when the current B-frame is accepted, it is preferred that the B-frame be dropped.

POB and MPFD schemes are similar because both drop low-priority frames only if a high-priority frame is going to be dropped otherwise. However, MPFD drops low-priority frames when they arrive at the buffer, while POB drops them when they are already queued in the buffer. PBS and MPFD are also similar because both drop frames when they arrive; however, PBS is not accurate in determining when a B-frame should be discarded because its decision is based on a fixed threshold in the buffer and not on the arrival pattern of

frames.

To efficiently implement the concept of the virtual buffer, we define a dynamic threshold for each frame type in the GOP structure. The threshold value is computed so that dropping higher-priority frames is avoided if possible. The threshold value is dynamically computed at the network node upon frame arrival by using information sent from the video source about the frame sizes of the frames within the future look-ahead period. A frame is dropped when it arrives at the buffer if the buffer occupancy exceeds its corresponding dynamic threshold value. Dropping frames upon arrival is fast and simple as compared to other algorithms that require searching the buffer for an appropriate frame to discard.

We choose B-frames as the first to be dropped when the buffer becomes congested. The B-frame is the lowest-priority frame in the frame dependency hierarchy for MPEG-2 traffic¹ and losing it will not affect decoding another frame at the destination. The spread of B-frames in the GOP structure causes B-frame loss to be distributed throughout the sequence and less likely to be noticed by the end user. Also, the frequency of occurrence of B-frames makes them readily available for dropping.

P-frames are usually dropped mainly to protect the following I-frame and other future P-frames that have lower sequence number (or higher priority). In general, all P-frames are subject to be dropped for the protection of the following I-frame. Dropping the P-frame with the highest sequence number (i.e., last P-frame in a GOP) will not cause another P-frame drop. In the next subsection, we describe the MPFD algorithm using the dynamic threshold with no jitter or losses in the incoming video traffic. Handling jitter and losses is discussed in Appendix 5.4.

Before we present the details of the MPFD algorithm, we stop to outline the three basic stages in the MPFD approach:

- The source stamps each frame with the frame sizes of the following L anchor frames and the frame interarrival time².

¹The priority assignments can be defined according to the traffic characteristics. MPEG-2 traffic is only used as an example in this work.

²Including full information in each packet may be redundant, however, we will assume first that each frame is stamped with all the information needed for its admission at the network node. Then, we will

- When the frame arrives at a network node, the stamped information is extracted from the frame header and then used along with the current buffer occupancy and the output rate to construct the virtual buffer. As will be shown in the next subsection, the virtual buffer construction is reduced to the evaluation of some simple equations to compute the dynamic threshold value.
- The computed dynamic threshold will indicate if the current frame can be admitted to the buffer or if it has to be discarded. If admitted to the buffer, the frame is enqueued and transmitted to the next network node.

In the next section, we elaborate on our work presented in [5] and show how the virtual buffer construction is emulated through an appropriate set of equations. We discuss deployment considerations for MPFD and the modifications required to handle jitter and loss in Section 5.4.

5.3 MPFD Algorithm

We define a dynamic threshold, T_{XY} , for each frame type in the GOP structure, where X is the frame type whose admission to the buffer is restricted by that threshold and Y is the furthest future higher-priority frame to be protected. This means that T_{XY} is defined such that X and Y and all high-priority frames between X and Y can be admitted to the buffer if the current buffer occupancy is less than T_{XY} . In other words, T_{XY} is the highest current buffer occupancy that guarantees no dropping of higher-priority frames in the virtual buffer. The threshold value depends on the future frame sizes that need to be protected. For example, when a B -frame arrives at the buffer, the threshold value, T_{BAL} , is calculated to determine whether the current available buffer space is enough to accept the future L anchor frames and the arriving B -frame³. If the buffer occupancy is below T_{BAL} , the B -frame is accepted, otherwise it is discarded. T_{BAL} should be carefully selected so that all L anchor frames are accepted. Other future B -frames might need to be discarded as well for the virtual buffer to accommodate the anchor frames.

discuss in a later subsection several ways of reducing the amount information sent with each packet.

³We will refer to L as the number of *look-ahead steps* for the dynamic threshold for the rest of the section.

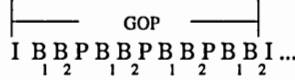


Figure 25: GOP frames assignments.

Let the GOP size be N frames and the distance between anchor frames be M frames. These video frames are labeled as in Figure 25. B-frames are labeled as B_1 and B_2 because their corresponding threshold value computation differs slightly, as will be shown later in this subsection.

Before computing T_{XY} , we define another threshold value, T_X^Y , as the threshold imposed on frame X to guarantee the acceptance of frame Y only. It is different than T_{XY} because it does not necessarily guarantee acceptance of high-priority frames between X and Y .

Let r_o be the output service rate and μ_X be the size of frame X in cells⁴. Now, we can compute $T_{B_1}^{A_i}$, $T_{B_2}^{A_i}$, and $T_P^{A_i}$. A_i represents the i -th anchor frame to arrive after the current frame for which we are computing the threshold. For the virtual buffer to accept the i -th anchor frame, there should initially exist enough available buffer space to start with to buffer any burst while receiving the i -th anchor frame, given that earlier frames do not overflow the virtual buffer. In other words, the buffer occupancy should be small enough to accommodate any bursts during that period. The maximum starting buffer occupancy (just before accepting frame X) is given by $T_X^{A_i}$, as shown in equation (29), where t_i is the time period during which the buffer receives both frames X and A_i , and μ_i is their frame sizes. This equation means that, given an output rate r_o , if the buffer occupancy is below $T_X^{A_i}$, the buffer can accept μ_i cells during t_i without overflowing.

$$T_X^{A_i} = B - \mu_i + t_i \times r_o \quad (29)$$

By using equation (29), we can compute the thresholds for each frame type by computing the appropriate corresponding μ_i and t_i . Equations (30)-(32) show the values for $T_{B_1}^{A_i}$, $T_{B_2}^{A_i}$, and $T_P^{A_i}$ with their corresponding μ_i and t_i values, where t_f is the frame interarrival time, t_X is the burst time for frame X , $t_X \leq t_f$, and $S(x)$ is the sequence number of frame x .

⁴Frames are segmented into fixed-size packets, which we will call cells.

$$\begin{aligned}
T_{B_1}^{A_i} &= B - \mu_i + t_i \times r_o \quad \forall i \in \{1, 2, \dots, L\} \\
t_i &= (2 + (i - 1)M) \times t_f + t_{A_i} \\
\mu_i &= \mu_{B_1} + \sum_{l=1}^i \mu_{A_l}
\end{aligned} \tag{30}$$

$$\begin{aligned}
T_{B_2}^{A_i} &= B - \mu_i + t_i \times r_o \quad \forall i \in \{1, 2, \dots, L\} \\
t_i &= (1 + (i - 1)M) \times t_f + t_{A_i} \\
\mu_i &= \mu_{B_2} + \sum_{l=1}^i \mu_{A_l}
\end{aligned} \tag{31}$$

$$\begin{aligned}
T_P^{A_i} &= B - \mu_i + t_i \times r_o \quad \forall i \in \{1, 2, \dots, \frac{N}{M} - 1\} \\
t_i &= (iM) \times t_f + t_{A_i} \\
\mu_i &= \mu_P + \sum_{\forall l: S(A_l) \leq S(A_i)} A_l
\end{aligned} \tag{32}$$

Notice that when $T_P^{A_i}$ is computed in equation (32), future B-frames are not accounted for because $T_{P A_i}$ determines only if there is a need to drop the current P-frame to protect a future frame of a higher priority given that all possible B-frames are dropped.

Since each anchor frame within L -look-ahead steps is guaranteed to be accepted, the threshold value for L -look-ahead steps has to be equal to the smallest i -look-ahead threshold for that frame. In other words, $T_{X A_L}$ is the minimum value of all $T_X^{A_i}$, $\forall i \in \{1, 2, \dots, L\}$ to guarantee acceptance of all L anchor frames. $T_{B_1 A_L}$, $T_{B_2 A_L}$, and $T_{P A_L}$ are shown in equations (33), (34), and (35), respectively.

$$T_{B_1 A_L} = \min(T_{B_1 A_i}) \quad \forall i \in \{1, 2, \dots, L\} \tag{33}$$

$$T_{B_2 A_L} = \min(T_{B_2 A_i}) \quad \forall i \in \{1, 2, \dots, L\} \tag{34}$$

$$T_{P A_L} = \min(T_{P A_i}) \quad \forall i \in \{1, 2, \dots, \frac{N}{M} - 1\} \tag{35}$$

In addition to the above threshold values, another threshold is defined on each frame type to prevent an overflow from happening in the middle of the frame reception. By introducing this threshold, the MPFD scheme will guarantee acceptance of complete frames.

The threshold, T_X , is given in equation (36). The network node algorithm to compute the threshold for each frame is given in Figure 26, where μ_c and t_c are the current frame size and its burst time, respectively.

As shown in the previous equations, the threshold is a function of the output rate, buffer size, and frame sizes. The actual sizes for corresponding future frames are sent from the sender to the network node, either in packet headers or in separate packets. Since the threshold computation is a function of the service rate and buffer size, this algorithm adapts to the current varying network condition. If the network becomes congested and the buffer's service rate decreases, this algorithm would still be effective in protecting the anchor frames over B-frames. If the network is not congested and the output service rate increases, the threshold value will increase, allowing more B-frames to be accepted, which will not happen in other fixed-threshold dropping schemes such as TBS and PBS [17].

$$T_X = B - \mu_X + t_X \times r_o \quad (36)$$

5.4 *MPFD with Jitter and Frame Loss*

5.4.1 Network Jitter

In Subsection 5.3, we derived the value for the dynamic threshold assuming fixed frame boundaries. In this subsection, we study the effect of jitter on the computation of the threshold values. Because of the jitter and queueing delay that the network introduces, packets may arrive spaced out or back-to-back. This behavior will certainly affect the performance of MPFD.

Consider a back-logged buffer at network node j , and let the buffer occupancy at that node be b . Let us assume that the buffer contains n video frames. Since those n frames are already in the buffer, they will be transmitted back-to-back with a rate equal to the buffer output rate. Since video frames vary in size, their frame boundaries vary accordingly. When considering jitter, we cannot assume fixed frame boundaries of future frames because this will produce inaccurate presentation of the virtual buffer, which will result in wrong

When a frame is received:

```

 $T = B - \mu_c + t_c \times r_o$  // Threshold to accept current frame
if (frame type =  $B_1$ -Frame)
    for all  $i < L$ 
         $t_i = (2 + (i - 1)M) \times t_f + t_{A_i}$ 
         $\mu_i = \mu_c + \sum_{l=1}^i \mu_{A_l}$ 
         $T = \min(T, B - \mu_i + t_i \times r_o)$ 
else if (frame type =  $B_2$ -Frame)
    for all  $i < L$ 
         $t_i = (1 + (i - 1)M) \times t_f + t_{A_i}$ 
         $\mu_i = \mu_c + \sum_{l=1}^i \mu_{A_l}$ 
         $T = \min(T, B - \mu_i + t_i \times r_o)$ 
else if (frame type =  $P$ -Frame)
    for all  $i < \frac{N}{M} - 1$ 
         $t_i = (iM) \times t_f + t_{A_i}$ 
         $\mu_i = \mu_P + \sum_{\forall l: S(A_l) \leq S(A_i)} \mu_{A_l}$ 
         $T = \min(T, B - \mu_i + t_i \times r_o)$ 
if ( $b > T$ )
    Drop(current frame and all future dependent frames)
else
    Accept(current frame)

```

Figure 26: MPFD algorithm

dynamic threshold values. To accurately compute the dynamic threshold in node $j + 1$, the node needs to know the arrival time of all future anchor frames that are included in the threshold computation so that the actual frame boundaries are determined. In the rest of this subsection, we define MPFD with jitter compensation algorithm, MPFD-JC, to solve the jitter problem.

In MPFD-JC, a state of timing information about future frames is maintained at the network node and updated each time a new frame arrives. The updated timing information is passed at the node buffer to the frame that is about to leave the buffer. For example, let us consider network node j with current buffer occupancy b . Assume that the first packet of an anchor frame, A_i , arrives at the buffer. Let us also assume that the first packet of a video frame, F_j , is about to leave the buffer. By knowing the buffer occupancy and the output rate, r_o , the interarrival time between F_j and A_i , t_{a_i} , is calculated, as shown in equation (37). The interarrival time for A_i can be sent with F_j to network node $j + 1$ and be used to calculate MPFD dynamic threshold at node $j + 1$.

The network node maintains the timing information for each anchor frame in the buffer. Timing information for an anchor frame, A_i , is mainly the buffer delay experienced before starting the frame transmission, denoted as t_{a_i} . Interarrival time for each received frame is computed according to equation (37) and then used to update the timing information. Each time a packet leaves the buffer, t_{a_i} will be decremented by t_{packet} , which is the packet transmission time. For each departing frame, the timing information is included in the frame header. *Note that the buffer contents are not accessed in the process of updating the interarrival times.* Information is gathered at frame arrival and is written at frame departure, i.e., when a packet arrives or leaves the buffer.

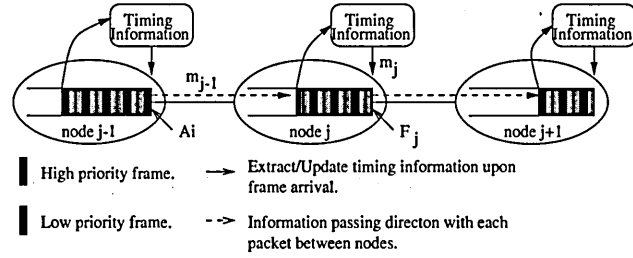


Figure 27: Passing interarrival information to past frames.

$$t_{a_i} = b \times r_o \quad (37)$$

For MPFD-JC with L -look-ahead steps, the buffer size may not be large enough to accommodate L anchor frames and all B-frames among them. Let us assume that the buffer has only $m_j < L$ anchor frames, we can obtain the interarrival times for the remaining $L - m_j$ frames from the last arriving frame (A_i in the above example) to node j from node $j - 1$. Since the arriving frame to node j carries interarrival times for the following m_{j-1} at the least, this information can be passed to F_j , as shown in Figure 27. After a frame traverses a number of network nodes, arrival information about the following L anchor frames will potentially be obtained.

Now, for the L -look-ahead step MPFD-JC, each network node will maintain timing information state that contains L values corresponding to the L anchor frames. Each value represents the time a frame needs before it starts its transmission out of the buffer. The

timing information state will be updated upon packet departure (by subtracting t_{packet} from each value) or frame arrival (by updating the interarrival times that are carried in the incoming frame header). It is noted that when links are congested, interarrival time information are updated faster due to larger buffer occupancies in the congested nodes. MPFD can be simply modified to account for network jitter. Equations (30)-(32) stay the same except for t_i value. The values for t_i in equations (30)-(32) are redefined in equation (38), where t_{a_i} is the interarrival time between the current frame and A_i . The pseudo code for MPFD-JC algorithm is given in Figure 28, where $FH_t[i]$ is the interarrival time between the current frame and A_i that is carried in the current frame header.

$$t_i = t_{a_i} + t_{A_i} \quad (38)$$

```

When frame  $j$  is received:
     $t_{a_j} = b \times r_o$ 
    for all  $j \leq i < L$  // Update timing information.
         $t_{a_i} = FH_t[i] + t_{a_j}$ 
     $T = B - \mu_c + t_c \times r_o$ 
    if (frame type =  $B_1$ -Frame or  $B_2$ -Frame)
        for all  $i < L$ 
             $t_i = t_{a_i} + t_{A_i}$ 
             $\mu_i = \mu_c + \sum_{l=1}^i \mu_{A_l}$ 
             $T = \min(T, B - \mu_i + t_i \times r_o)$ 
    else if (frame type =  $P$ -Frame)
        for all  $i < \frac{N}{M} - 1$ 
             $t_i = t_{a_i} + t_{A_i}$ 
             $\mu_i = \mu_P + \sum_{\forall l: S(A_l) \leq S(A_i)} \mu_{A_l}$ 
             $T = \min(T, B - \mu_i + t_i \times r_o)$ 
    if ( $b > T$ )
        Drop(current frame and all future dependencies)
    else
        Accept(current frame)

When a packet is transmitted:
    for all  $i < L$ 
         $t_{a_i} = t_{a_i} - t_{packet}$ 

For each departing frame header:
    for all  $i < L$ 
         $FH_t[i] = t_{a_i}$ 

```

Figure 28: MPFD-JC algorithm

The complexity involved in maintaining a state of timing information is not significant when compared to the buffer maintenance complexity in POB scheme. The state information may only correspond to an array of L variables, where L is a small number. In our simulation evaluations, $L = 10$ was sufficient to outperform all existing dropping schemes.

5.4.2 Frame Loss

The MPFD algorithm depends on the source for providing network nodes with information about future frames to create the virtual buffer. The virtual buffer is created correctly at each node if we assume that there is no anchor frame loss in the upstream nodes because the information carried in a frame will correctly describe future frames within the look-ahead period. However, anchor frames may be discarded resulting in incorrect or not updated frame information carried by previous frames. To illustrate this problem, let us consider a scenario of a B-frame that carries information about a later anchor frame, A_i . The B-frame is accepted and buffered at network node j . Assume that A_i is dropped when it arrives at node $j - 1$. Now the B-frame will have incorrect information about a dropped frame. The B-frame proceeds to node $j + 1$ where it gets dropped to avoid dropping A_i . The B-frame is dropped because of incorrect or not updated information about A_i causing more frame drop than needed.

Therefore, anchor frame loss needs to be updated in previous frames that are still in the physical buffer to avoid incorrect dropping of frames. To solve this problem we follow a method similar to MPFD-JC and define an extended MPFD algorithm with loss compensation, MPFD-LC. In MPFD-LC, the network node maintains information state about anchor frames sizes in the look-ahead period and updates this information with any changes carried in incoming frames. The state includes L frame sizes that correspond to the L anchor frames in the look-ahead period. If a node drops an anchor frame, it updates its corresponding maintained frame size with a zero. The node updates the information carried in each departing frame with any frame drop. A pseudo code for the MPFD-LC algorithm is shown in Figure 29.

Note that dropping a B-frame does not require any update because B-frame information

```

When frame  $j$  is received:
  for all  $i < L$  // Update size information.
     $\mu_{A_i} = FH_\mu[i]$ 
    if anchor frame
      if frame dropped
         $\mu_{A_0} = 0$ 
For each departing frame header:
  for all  $i < L$ 
     $FH_\mu[i] = \mu_{A_i}$ 

```

Figure 29: MPFD-LC algorithm

is not carried in previous frames and is not used for look-ahead. Anchor frame loss is rare because a loss will first occur in B-frames; therefore, updating information about lost anchor frame is infrequent and should not introduce significant overhead.

5.5 Deployment Considerations

Thus far we have assumed that each frame header contains frame sizes of all L future anchor frames. This will create redundant information with a high overhead that is carried on the network⁵. In this section, we will address different ways in which the source can provide the network nodes with the necessary information but with lower overhead.

One way of sending information from the source to the network node is to provide the network node with incremental information in each frame header. In this method, each frame header will contain the frame size of the L -th anchor frame that follows that frame. Each network node will maintain an explicit state that contains L data items, and each item will contain the frame size of an anchor frame within the look-ahead period. When a frame arrives, the L -th anchor frame size is extracted and updated in the state. Since the distance between anchor frames (M) is typically three (see Figure 25), anchor frame size information can be sufficiently sent every three frames; however, this information can be duplicated in each of the three frames. If a frame is dropped in a network node, then subsequent nodes will not be updated with frame information that is carried in that frame,

⁵Although, the look-ahead step required to match the performance of a complex algorithm such as POB is relatively small (4 to 6 in our evaluations).

which will affect the functionality of MPFD. To overcome this problem, the information that was carried in the dropped frame can be appended to the information carried in the next frame header. Maintaining a state of L values⁶ does not add a significant overhead to the network node and it is much simpler than manipulating the buffer contents, which happens in POB scheme.

The choice of the method of sending information to the network depends on two main parameters that govern the amount of information sent in the frame header: available bandwidth and router capabilities. If MPFD is implemented on routers with limited capabilities, then stamping full information in each frame will eliminate maintaining state at the router at the expense of redundancy in sent information. At the other extreme, routers with sufficient capabilities can maintain state that contains future frame sizes to minimize the bandwidth overhead. We have seen that the MPFD performs better than other threshold-based schemes. The dynamic threshold value makes this algorithm adaptive to the changes in the video stream burstiness and utilizes the buffer space between B-frames and anchor frames. The MPFD requires the knowledge of future incoming frame sizes. This information needs to be transmitted to the network node from the video source. Hence, the sender has to support the MPFD scheme in its operation. Information about future frames can be included in packet headers or sent on separate periodic packets. Either way, sending information about frame sizes adds bandwidth overhead to the established connection.

Because of packet loss in the network, packets that carry information about future frames may get lost. In that case, the MPFD has to replace this information with an estimate. The importance of the lost information varies according to the frame type it belongs to, and the distance between the packet that carries the information and the frame the information is about. Also, if the look-ahead is short then if some information is lost, the acceptance decision may not be critical after the frame drop because the buffer is likely to have more space after that drop.

The general MPFD can be applied for video-on-demand streaming to enhance video quality because information about future frames is available. In real-time video, however,

⁶In our simulation evaluations, a reasonable value of L is 10.

this type of information is not available. Therefore, MPFD can be limited to short look-ahead distance and utilizing information about a small number of future frames.

5.6 Simulation and Results

To study the performance of MPFD, we consider a network node with a simple architecture. Since each flow in the node has a designated buffer space and output rate, we can look at one video flow in the simulation, as shown in Figure 7. When a packet arrives at the buffer, a decision is made on whether or not to accepted it depending on the buffer management scheme applied. The video traffic model used is similar to the model discussed in 4.7.

5.6.1 Metrics

Part of the network node intelligence is the ability to distinguish between video frame types. This allows the network node to implement frame dependency drop, i.e., when a frame is dropped, all of its dependent incoming frames are dropped as well. By using frame dependency in dropping frames, only usable frames will be passed to the next network node, which will reduce the network load and cause some congestion situations to be eliminated. Dependency dropping is used in this work to evaluate the performance in terms of goodput.

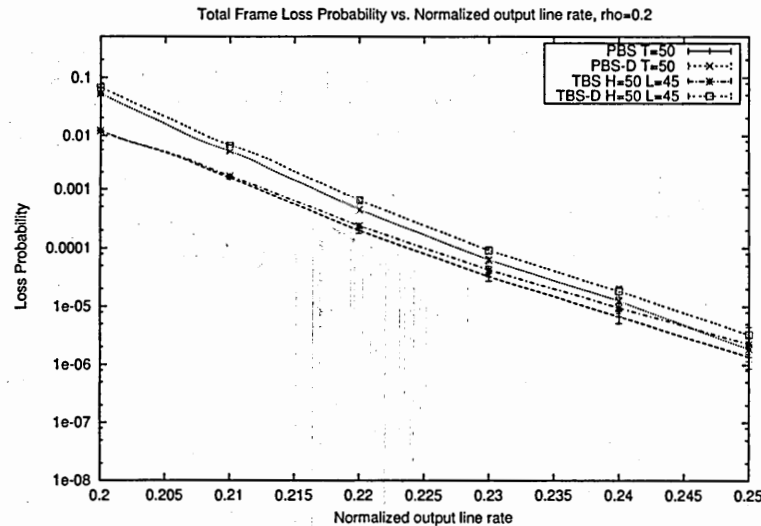


Figure 30: Total loss.

Furthermore, we define $P_{loss} = 1 - \text{goodput}$; we choose these definitions of goodput and loss probability because they are more representative measures of the integrity of the end users' perceived video quality than other measures [61, 20]. MPFD is designed to consider the video traffic dependency among frames. However, PBS and TBS were not designed for video traffic that has dependency among frames. Since we are measuring the goodput (or the usability of transmitted video frames), it will not be fair to compare MPFD with the PBS and POB as they are because their goodput performance will be low. Therefore, we implemented modified PBS, TBS, and TD schemes that are able to perform dependency dropping; we denote the modified versions as PBS-D, TBS-D, and TD-D, respectively.

The goodput performance of PBS-D and TBS-D is better than the original PBS and TBS schemes, as shown in Figure 30. Since the corrupted or incorrectly decodable frames are dropped upon arrival, the buffer will have more space to accept usable frames, thereby increasing the goodput.

5.6.2 Results

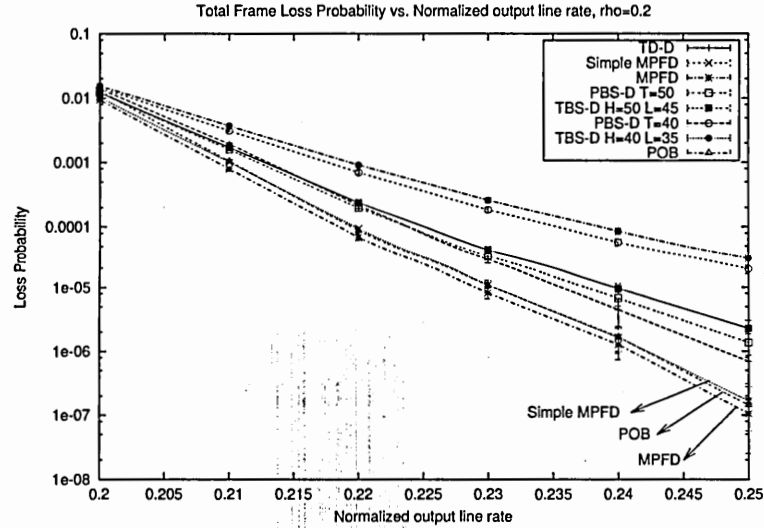


Figure 31: Total loss for 20% load.

We compared the results of MPFD with four dropping schemes: TD-D, POB, PBS-D, and TBS-D algorithms. In Figures 31–34, we plotted cell loss probability for all dropping

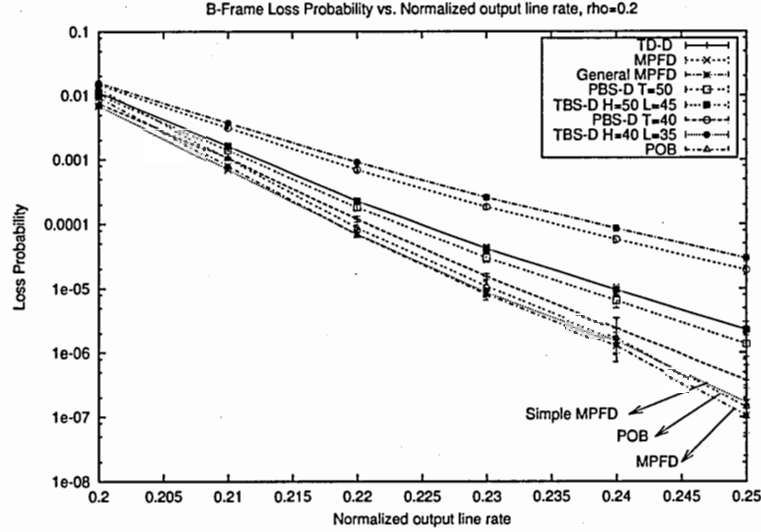


Figure 32: Loss in B-frames for 20% load.

schemes against the normalized output line rate for a load of 20% and with a confidence of 90%. Figures 35–38 show similar results for a load of 23%. In addition to the results for MPFD with $L = 10$, we plotted results for simple MPFD with $L = 1$ to show the lower performance bound of MPFD.

Figure 31 shows the total loss probability for all dropping schemes. The total loss of complete frames in both MPFD and simple MPFD appears to be the smallest of all threshold based schemes while it achieves a comparable performance of POB scheme. This indicates that the MPFD scheme results in better bandwidth utilization than TD-D, PBS-D and TBS-D. Similar results are also shown in Figure 35. Notice that, with simple MPFD, total loss is significantly reduced, which results in better video quality. The reason behind this significant improvement can be seen in Figures 32 and 36, where we can see that the reduction in the total loss is mainly caused by a significant reduction in B-frame loss. This loss drop is one advantage of the dynamic threshold in MPFD over the fixed threshold(s) in PBS-D and TBS-D.

Notice that the performance of PBS-D and TBS-D is governed by the appropriate selection of the threshold value. The B-frame loss in PBS-D is increased by an order of magnitude when the threshold value is changed from 50 to 40. Therefore, the threshold

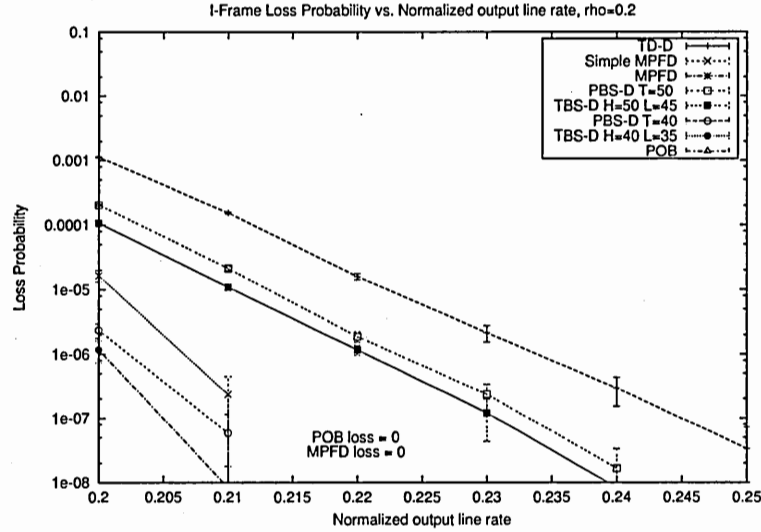


Figure 33: Loss in I-frames for 20% load.

value has to be chosen carefully to meet the burstiness of the video stream. In MPFD, however, it is not required to define a fixed optimum value for the threshold.

Figures 33 and 34 show the cell loss probabilities in I-frames and P-frames, respectively. The simple MPFD results show a loss reduction in I-frames and P-frames and better performance than TD-D, but not as good as other threshold-based dropping schemes. In MPFD, however, the I-frames encountered no loss and the P-frames had only 3.5×10^{-6} , which is the lowest loss probability among all dropping schemes, including POB. Similar results are shown in Figures 37 and 38 for a load of 23%. The general MPFD performs the best among the other dropping schemes. It is important to notice that, for comparable results of anchor frame loss, the corresponding B-frame loss in MPFD is one to two orders of magnitude less than that for TBS-D and PBS-D.

The simple MPFD has shown a significant decrease in loss probability in I-frames and P-frames without increasing loss probability in B-frames. With knowledge of only one future anchor frame size, it provided I-frame loss reduction better than PBS-D with $T = 50$, and TBS-D with $H = 50, L = 45$. The loss in P-frames is comparable to PBS-D with a threshold value of 50 for a normalized output rate greater than 0.23, which is equal to the average input traffic rate as shown in Figure 38. Since simple MPFD represents the lower

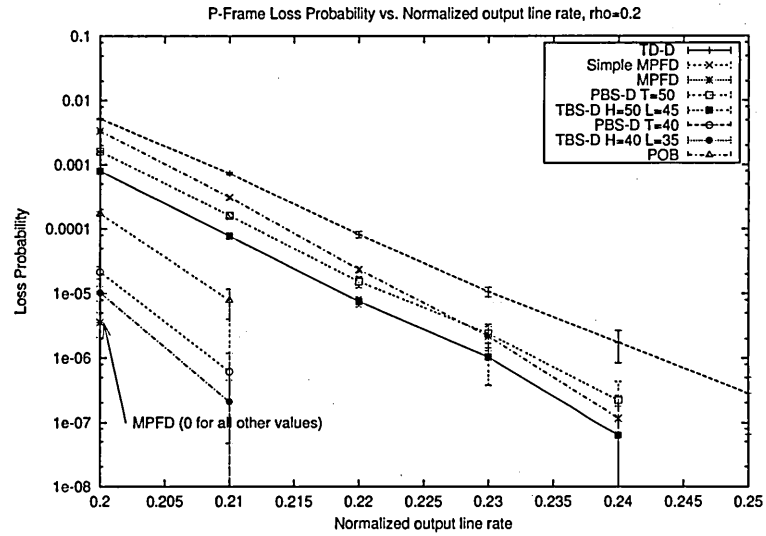


Figure 34: Loss in P-frames for 20% load.

performance bound for MPFD, the results show that MPFD is efficient in protecting high priority frames while increasing the usability of the video frames.

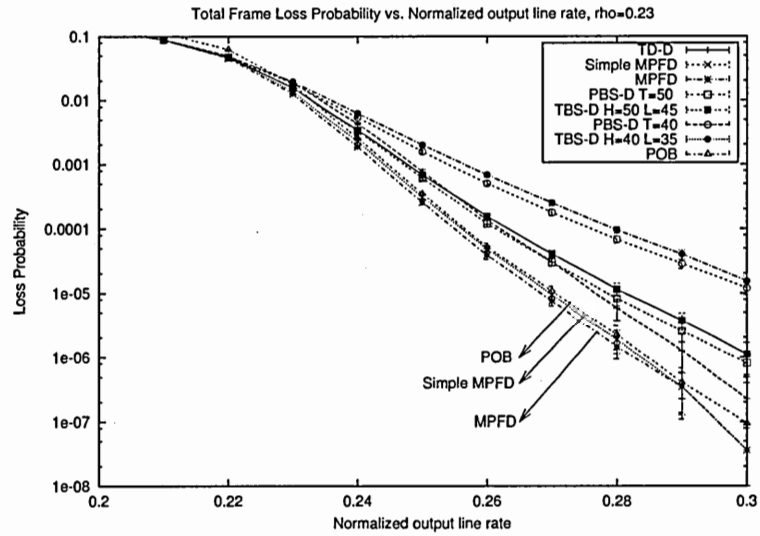


Figure 35: Total loss for 23% load.

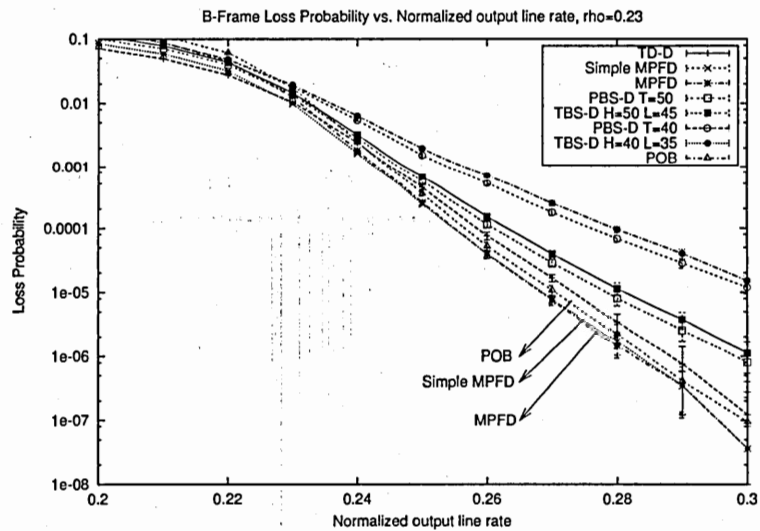


Figure 36: Loss in B-frames for 23% load.

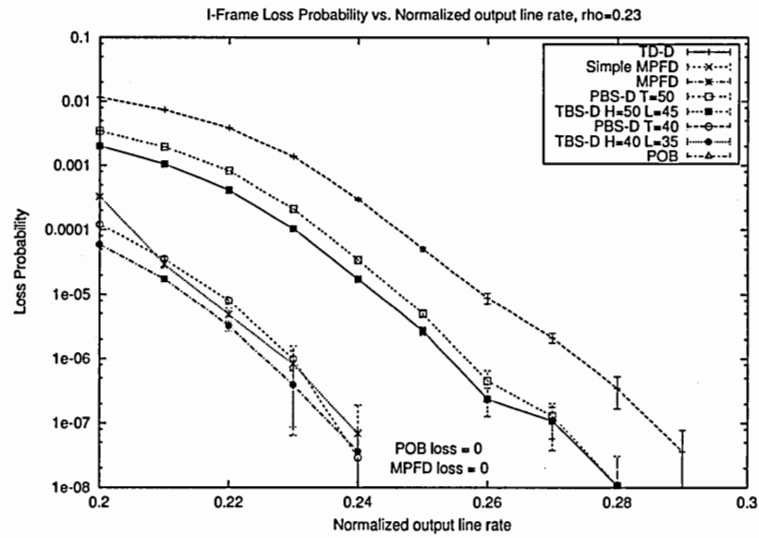


Figure 37: Loss in I-frames for 23% load.

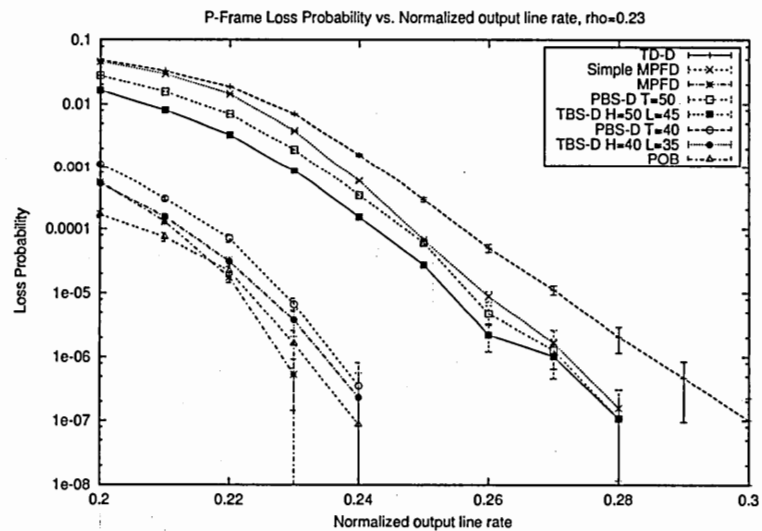


Figure 38: Loss in P-frames for 23% load.

CHAPTER 6

UTILITE: UTILITY AWARE RATE ALLOCATION

6.1 *Introduction*

With the large number of traffic flows that traverse network nodes, congestion and packet drop are likely to happen at network routers. Even though end-to-end congestion control mechanisms are applied to have some flows adapt to the current network congestion condition, some network flows may be more aggressive in seeking more bandwidth than other flows, some other flows may be ill behaved and may not apply congestion control at the end system, which results in having them dominating a network links. To overcome this problem, rate allocation mechanisms are applied at network routers to enhance the performance of end-to-end congestion mechanisms by protecting well-behaved flows from aggressive, ill-behaved flows. Flows are allocated bandwidth similar to other flows so that the effect of ill-behaved and aggressive flows is contained. Hence, rate allocation mechanisms are important for the network to provide all users with the same service level.

Rate allocation mechanisms can vary in many ways in their performance and how they provide network flows with their share of network resources. We identify two fundamental characteristics by which they are classified.

- **Scalability:** Since rate allocation mechanisms inherently manage a large number of network flows, it is important that their complexity does not become a performance bottleneck as the number of flows in the network increases. Basing rate allocation mechanisms on a per-flow framework, for example, will encounter scalability issues. A network router applying a per-flow mechanism has to classify packets into flows, update flow state variables, and perform operations based on the per-flow state, which limits its scalability.
- **Rate allocation criteria:** A key difference between rate allocation mechanisms is

the criterion they follow to allocate rate to flows. These criteria are determined by many factors such as pricing model, application class, and application characteristics. At a high level, we classify them into two categories: rate-based and utility-based. In rate-based allocation mechanisms, flows sharing a network link are allocated a share that can be the same for all flows; or can be different when rate allocation depends on a weight that represents the relative bandwidth allocation of the flow with respect to other flows [55, 53, 45, 39]. In utility-based allocation, rate is allocated with respect to the application-level QoS of the flow. Since different network applications have different rate requirements to achieve a certain QoS level, the fair rate allocated to network flows may differ accordingly. Thus, flows traversing a network router can be provided a share of the link capacity that results in similar utility as other flows in that link [13, 51, 2].

Utility fairness mechanisms are presented in the literature [13, 37]. However, these proposed mechanisms are achieved by using per-flow mechanisms or focused on maximizing the aggregate utility rather than providing utility fairness. Maintaining a state per flow is undesirable because of complexity and scalability issues involved especially at the core routers [55].

It is evident that rate allocation that is utility-based better represents the fairness with regard to the end applications than rate-based allocation. Therefore, utility becomes key component in rate allocation, especially when applications with diverse QoS requirements are sharing the network. Since rate allocation mechanisms inherently reside in the core network routers, we consider simplicity and scalability of the router significant to its performance and need to be considered in rate allocation mechanisms. We, therefore, propose Utilite, a scalable utility-based fair rate allocation architecture that can approximate the utility max-min fairness, a fairness approach that will be explained in the next section.

The proposed architecture uses utility information transmitted from the sources to the network routers to allocate rates to flows such that utility fairness is achieved. Although utility-based max-min allocation schemes are present in the literature, these schemes are not scalable. The Utilite architecture is scalable since it does not maintain per-flow state.

Instead, it leverages the concept of dynamic packet state [56] to realize scalability.

In the next section, we first present an overview about utility functions and their properties. We then present the definition of max-min utility fairness that is used in this work.

6.2 *Utility-Based Fairness*

6.2.1 Utility Functions

The application perceived QoS was discussed in [13, 52] in terms of utility functions, and several different shapes for utility functions that describe different classes of applications were presented. Traditional data applications such as file transfer, electronic mail, and remote terminal access can tolerate end-to-end delays, and therefore, this class of applications are called “elastic traffic”. A convex utility function like that in Figure 39 can characterize the utility function for elastic traffic.

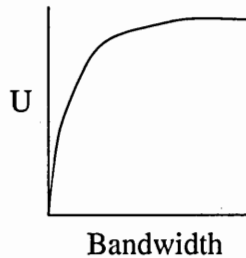


Figure 39: Utility corresponding to rate-based rate allocation

Delay-sensitive applications such as real-time traffic generally have hard real-time requirements. Examples of delay-sensitive applications are video streaming, traditional telephone, and interactive applications. In this class of applications, the performance will stay constant as long as the packets arrive the destination within the delay bound. It would not matter if the packet arrives early, but the performance drops sharply if the packet arrive beyond this bound. The utility curve for such applications is typically a single-step function as shown in Figure 40.

Another class of applications are called “rate-adaptive”. These applications can adjust their transmission rate according to the network congestion level. In this case the performance of the application solely depends on the signal quality. At high bandwidth, the performance degradation is small because the signal quality is higher than what humans

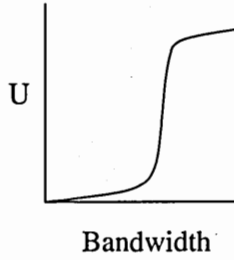


Figure 40: Utility corresponding to rate-based rate allocation

need. Similarly, at low bandwidth the marginal utility is small because the signal is already of low quality. The utility function for this class of applications takes the shape of that in Figure 41.

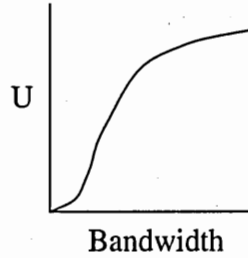


Figure 41: Utility corresponding to rate-based rate allocation

6.2.2 Motivation

When various application classes share the network, it becomes evident that the application utility is affected by the rate allocation scheme provided by the network. To illustrate this effect, we consider an example of two applications: FTP application and video streaming application. The former represents elastic traffic class while the later represents the delay-sensitive class. The utility functions for both applications are shown in Figure 42, where (C1) is for video streaming traffic and (C2) is for elastic traffic. The utility is represented by a value between 0 and 1, where 1 represents the highest utility or 100% satisfaction.

Let us consider a scenario of these two applications sharing a congested network link with rate-based allocation mechanism that provides both applications with a rate r_a , as shown in Figure 42. We notice that, even though they share equal network resources, there is a large difference between their operating utility values. The FTP application is enjoying a high application performance while the video application performance is lacking bandwidth.

Unfairness to the video application is clear as it requires more bandwidth to achieve the same performance as the FTP application¹. To grant the video application more bandwidth, we consider applying weighted rate-based fair allocation schemes because they can provide relatively more bandwidth to one application than another according to relative weights that are given to each application. In these schemes, the higher the weight an application has with respect to other weights, the more bandwidth it gets. If weighted rate allocation is used, however, the video application can be assigned higher rate to increase its utility. Unfortunately, a fixed weight assignment for different applications does not result in equal application utility at all times. This is due to the non-linear nature of utility functions. Applying a set of weights to achieve equal utility for all applications at certain level of network congestion will not work if the network congestion level changes. Hence, when flow fairness in utility (or application-level quality) is pursued, simple rate-based fairness mechanisms are not the appropriate solutions.

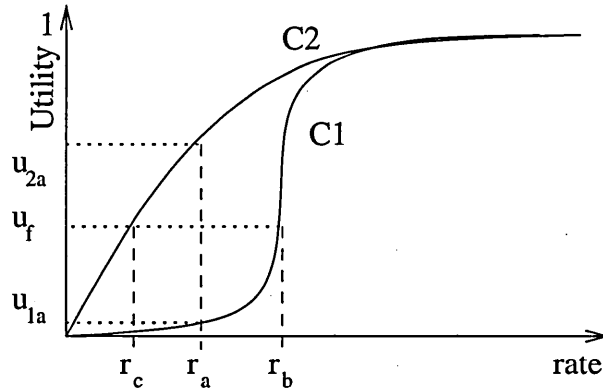


Figure 42: Utility corresponding to rate-based rate allocation

If rate allocation is based on utility, however, the flows sharing a link will enjoy the same level of quality even though they may have different rate allocations. In our previous example, instead of allocating r_a for both applications, the network can provide equal utilities for both applications irrespective to their different allocated bandwidth as long as the aggregate allocated bandwidth in the link does not exceed its capacity as shown in Figure 42. With equal utilities for both applications, the video application is allocated r_b

¹The formal definition for utility fairness will be presented later in this section. For sake of this discussion, however, we assume utility fairness when applications achieve equal utilities.

while the FTP application is allocated only r_c .

6.2.3 Utility fairness

So far we have discussed the importance of incorporating utility to achieve fairness in rate allocation mechanisms but did not address the fairness criterion to adopt in this work. A fairness criterion is defined by a welfare function that it tries to maximize. Two common fundametal fairness criteria are used in the literature to utility-based rate allocation:

- *Max-min Fairness:* Max-main fairness [7, 13] is a popular fariness concept in computer networks. The basic idea behind this criterion is to allocate network bandwidth that result in distributing utility as equally as possible such that each flow is allocated the maximum possible utility. Consequently, this criterion maximizes the smallest utility in the network. The welfare function thus becomes:

$$W(u_1, u_2, \dots, u_n) = \min(u_1, u_2, \dots, u_n) \quad (39)$$

- *Proportional Fairness:* The propotional fairness [33] is becoming increasinlg popular in the field of computer networks. In this criterion, a proportionally fair utility allocation vector u is one such that, for any other utility allocation vector u^* , the aggregate of proportional changes is zero or negative, as shown in equation (40). For a utility function in the form of $U = \log(r)$, propotional fairness reduces to a welfare function that is simply the sum of utilities, as given in equation (41). Therefore, it maximizes the overall network performance by achieving the maximum aggregate utility of the network flows.

$$\sum_{s \in S} \frac{u_s^* - u_s}{u_s} \leq 0. \quad (40)$$

$$W(u_1, u_2, \dots, u_n) = \sum_{i=1}^n u_i \quad (41)$$

Our focus in this work, however, is to allocate rate to flows such that each flow in the network enjoys the largest possible utility. Ideally, we would like for all flows to receive

the same utility. This falls under the definition of max-min allocation. Utility max-min (UMM) fairness is proposed in [13]. This fairness criteria seeks to maximize the utility the network provides to the applications by maximizing the minimum utility achieved by any application in the system.

Before we present the formal definition of utility max-min, we first define a utility-based ordering for rate allocation vectors. Let $A(a_0, a_1, \dots, a_{n-1})$ be a feasible allocation vector, where a_i is the bandwidth allocated for application i . Coupled with this vector, $\bar{A} = (a_{i_0}, a_{i_1}, \dots, a_{i_{n-1}})$ is defined as a utility-ordered vector such that $U_{i_k}(a_{i_k}) \leq U_{i_{k+1}}(a_{i_{k+1}}), \forall i, k \in [0, n-2]$ where U_i is the utility function for application i . Given any other bandwidth vector B and its utility-ordered vector $\bar{B} = (b_{j_0}, b_{j_1}, \dots, b_{j_{n-1}})$, then $A >_u B$ iff there exists some m such that $U_{i_k}(a_{i_k}) = U_{j_k}(b_{j_k})$ for $0 \leq k < m$ and $U_{i_m}(a_{i_m}) > U_{j_m}(b_{j_m})$.

Now we can define utility max-min allocation as the feasible bandwidth allocation vector that is largest under the ordering defined by $>_u$ [13].

In other words, a vector R is utility max-min fair if it is feasible and $U_i(r_i)$ cannot be increased while maintaining feasibility without decreasing $U_j(r_j)$ where $U_j(r_j) \leq U_i(r_i)$. This means that the minimum utility value in R is maximum among all feasible allocation vectors.

6.3 Ideal Utility Max-Min Rate Allocation Algorithm

In this section, we present an algorithm that achieves max-min fairness. It will open the road to explaining the Utilite scalable architecture and understand the approximations it contains. The main concept of the algorithm is to allocated bandwidth in a way that delivers max-min fair utilities to all flows traversing a network link.

The network model used is connection oriented so that all packets from the same flow follow the same path. A feedback mechanism will update the source with the allocated bandwidth to the flow in order to converge to it. The feedback is sent from the destination to the source periodically.

During connection establishment, the utility functions are sent to the network nodes to

be used by the algorithm. When congestion occurs, the utility functions are used to allocate appropriate rates to each flow based on max-min fairness criteria. The utility function for a flow can potentially change over time, therefore, a mechanism is required to update the network nodes with the most recent utility function of the flow. This can be done through signalling or on a per-request basis.

6.3.1 Ideal Algorithm

Consider a network link l with capacity C_l and a set of flows F_l . We define a subset of F_l , E_l , that contains flows with utility that exceeds the fair utility during congestion. We also define A_l as the total rate corresponding to remaining link bandwidth after subtracting all rates for flows with utility less than the fair utility. To find the fair utility, the algorithms follow three steps. It is initialized with $u_f = 0$ and $E_l = F_l$.

- At each link l , find temporary rate allocations for flows in E_l that results in saturating the link and achieving equal utilities:

$\forall i \in E_l$, allocated $t_{i,l}$ such that

$$\sum_{i \in E_l} t_{i,l} = A_l \text{ and}$$

$$\forall j \in E_l, U_j(t_{j,l}) = U_i(t_{i,l})$$

- The iterations stop when there is no change in the fair utility:

if $U_i(t_{i,l}) = u_f$ then stop, else

$$U_i(t_{i,l}) = u_f.$$

- Remove all flows with original operating utilities less than u_f :

$\forall i \in E_l$, if $U_i(r_i) < u_f$ then

$$E_l = E_l - \{i\}, A_l = A_l - r_i$$

This algorithm runs when there is a change in the flow rate, or when a flow joins or leave the link.

6.3.2 Packet Drop

After the fair utility for a link is calculated, the utility function for each flow is used to calculate its allocated fair rate. If the incoming flow rate is higher than the fair rate, some packets will be uniformly accepted according to the probability P_a given in (42).

$$P_a = 1 - \frac{r_i(u_f)}{r_i(u_i)} \quad (42)$$

6.4 Overview of Utilite

As mentioned earlier, simplicity and scalability of rate allocation mechanisms in the core routers are desired features in the network. When scalable, the router computational and allocation complexity does not increase with the number of flows traversing it. Until now, utility-based max-min fairness is performed on a per-flow manner that involves complexity and scalability hardships. When a router implements per-flow mechanisms, it has to classify incoming packets into flows, update the flow state variables, and perform certain operations based on the per-flow state. This becomes significant when the number of flows is large. Therefore, it is important for fair rate allocation mechanisms to be scalable, especially at the core routers.

Scalable mechanisms have been proposed for rate-based allocation [55], we project this notion of scalability on utility-based rate allocation. Scalable utility-based mechanisms are more complex to implement when compared to rate-based mechanisms due to their reliance on parameters other than the flow rate, such as utility function information.

For the rest of this section, we present the Utilite framework we are following, the challenges that need to be resolved, and the Utilite design elements.

6.4.1 Utilite Framework

The different elements of Utilite framework are:

- **Utility information delivery:** Utility curves are used as a basis for allocating bandwidth to network flows. Thus, network nodes would require utility function information from all flows traversing it to correctly allocate bandwidth. Since each application has its own utility function that is defined at the source, utility functions have to be reported from the source to network nodes along the flow's path.
- **Dynamic packet state (DPS):** The notion of DPS is used for information delivery from the source to all network nodes on the flow's path. In addition, DPS maintains state information for the flow that is updated in intermediate nodes when the flow state changes, then carries the updated state to subsequent nodes. Using DPS is a principal element in Utilite framework to achieve scalability since the flow state is carried in the packets rather than residing at each of the intermediate routers.
- **Router state:** At each intermediate router, aggregate information about all the flows traversing it should be maintained in a scalable fashion. The information size, and therefore, state size in the router, should not increase with the number of flows traversing it. This information is used in conjunction with the additional flow-specific parameters carried in packet DPS to approximate fair utility.
- **Router utility-based rate allocation:** After the max-min fair utility is found, the corresponding rate is computed using the utility function of the flow. If the flow rate is higher than the allocated rate, some packets will be dropped.
- **Feedback and rate adaptation:** Once the flow is allocated a certain bandwidth, the destination sends feedback to the source signaling the new rate. The source then adapts its rate to the new rate to avoid packet loss.
- **Bandwidth probing:** The fair bandwidth allocation may change according to the network dynamics and congestion. The source can be notified to lower its rate when its allocated rate is decreased, however, the source will have no knowledge when its allocated rate is increased. Therefore, the source will either probe the network bandwidth periodically to search for bandwidth, or depend on network feedback about

the available bandwidth.

6.4.2 Challenges

To realize the scalable design of utility-based rate allocation, many challenges arise because of the simplicity and the scalability sought at the routers. These challenges are:

- **Utility curve approximation:** since the utility function is used at intermediate routers to provide utility fairness, it is reported from the source to the routers by providing the flow utility information through DPS. Ideally, representation of the often non-linear utility functions should be flexible and general enough to include various types of utility functions. At the same time, it should be simple and feasible with respect to the related computations at the intermediate router. Therefore, there is a tradeoff between generality and simplicity in representing utility functions. For this purpose, utility functions are generally approximated using a set of curves that span their entire range [46]. We refer to these curves as “piece-wise curves”. The number of piece-wise curves used to represent a utility functions is determined by many factors such as the smoothness of the utility function, the linearity degree of the approximation curve, the utility range the curve represents, and approximation accuracy. These factors, coupled with the practical computation simplicity at the router, should be taken into account when approximating a utility function. For example, although non-linear curves tend to approximate a higher range of utility with more accuracy, it is harder to use in computations at the network routers.
- **DPS size:** To minimize the overhead introduced by including the DPS in the flow packets, the information conveyed from the source to the intermediate routers should exclude any redundant data that is not used given the current network condition. The conveyed information should only cover what is needed for the current network condition and its dynamics. For example, if a piece-wise utility curve composes sufficient information for the rate allocation mechanism to compute the fair utility, the rest of the utility function can be excluded from the conveyed information. The challenge is

how to optimally provide small but reasonably sufficient information to the network routers.

- **Information format and delivery:** After transmitting information about utility functions to the intermediate routers, this information is aggregated from all flows and kept at the router to be used to allocate the appropriate utility-based rate for each flow. In order to be scalable, the aggregate information has to be gathered from all flows without packet classification or per-flow processing. Since subsequent packets from the same flow may carry the same information in their DPS, any information aggregation must avoid information duplication without sacrificing scalability.
- **Packet loss:** During congestion, several packets will be discarded when a flow is allocated less rate than its nominal rate. Discarding packets and losing the DPS state in them, however, should not affect the functionality of the rate allocation mechanism at any of the intermediate routers, and any loss encountered in the transmitted DPS information should be compensated for. The compensation should most importantly guarantee the validity of the aggregate information kept at intermediate routers after a packet drop occurs.
- **Convergence:** It is important for the algorithm to converge to the optimal max-min fair utility allocation and be robust with regard to the network dynamics. One factor that can cause convergence instability is approximation error in the fair utility value, which, if becomes large, causes high oscillation of the allocated rate. The introduced oscillation will cause either excessive packet drop if the link capacity is exceeded, or low link utilization when the aggregate rate decreases below the link capacity. This inherently leads to performance degradation in the application QoS. Therefore, the utility values achieved through the algorithm should converge to the optimal value quickly.

Next, we discuss the design elements that are adopted in the algorithm to overcome the challenged presented in this section.

6.4.3 Design Elements

In this subsection, we will explain the design of Utilite and how it achieves scalable utility-based fairness. The design tradeoffs are also discussed.

One of the main design issues is how to represent the utility function of a source at the router. Initially, the user chooses the rate, and therefore, the utility it wants to operate on. Choosing a particular operating utility may depend on the available link capacity, the source budget, or the applications' current need. Instead of sending the complete utility function to intermediate routers, the source can send only a piece-wise curve in which the flow's operating utility lies. However, this part of the curve has to be sent in the packet header. To minimize the amount of information sent in the packet header, we approximate utility function by a piece-wise *linear* curve because a linear curve has the advantage of having less parameters to represent it and has a constant slope, which can significantly reduce the computational complexity of the Utilite architecture. Piece-wise linear approximation of utility functions is also used in [13]. In our experimental evaluation, a utility function can be approximated using four to eight linear curves, which is reasonable and results in acceptable utility range coverage per curve.

For every packet, a linear approximation of the part of the utility function in which the flow operating utility is sent. Unfortunately, under some congestion conditions, the change in utility may push the flow to operate outside the coverage region that is provided by the linear approximation of the utility function. At that point, the router will not have information about the utility function for that flow that represents the new operating utility. One solution would be to approximate the unknown part by linearly extrapolating the approximate line to increase its coverage range. However, this may result in a conflict with the utility function characteristics. For example, when a link becomes congested and the utility has to be reduced, the extension of a linear line may indicate a zero rate at a utility higher than the fair utility of the link. This in turn would indicate that the flow will be allocated a negative rate to achieve the fair utility, which is unrealistic. To avoid this problem, we coarsely approximate the unknown parts of the utility function to be consistent with the original utility function characteristics. As a safety net, we use two extreme points

on the utility function for this approximation. The first point represents the rate at which the flow reaches a utility of one and the second point is simply the point at which the flow reaches a zero utility. Therefore, in every packet, the router will eventually have a three line approximation of the utility function: one that is accurate, and two that are coarse approximates of the utility function but consistent with the general utility function characteristics. It is worth noting that the coarse approximation will become accurate once the source changes its operating utility to fall in the coarsely approximated region because it will then send an accurate line approximation for the new operating utility.

Utility function information are aggregated at the intermediate routers in a value that represents the relationship between the aggregate rate at the link and the fair utility. We choose such representation of aggregated information at the routers because any adjustment in utility is triggered by a change in the incoming aggregate traffic at the router. When congestion occurs, the router will be receiving traffic that saturates and exceeds its output link capacity. To eliminate congestion, the packets from different flows will be dropped according to their utility functions and the computed fair rate. Since the excess traffic rate can be measured at the router, the aggregate utility information at the router should facilitate using excess rate in calculating the adjustment needed for the current utility to reach the fair utility that will remove the congestion.

To represent the relationship between fair utility and aggregate rate, we ideally need to generate a new utility function for the aggregate link rate. More precisely, the rate of change in utility with respect to the change in the current aggregate rate is needed. To obtain this value, Utilite calculates the aggregate rate of change in the fair utility (slope) with respect to the aggregate link rate, namely $s = \frac{\Delta R}{\Delta u}$, where R is the aggregate link rate. This value can be used to reflect a sought reduction in aggregate rate into a reduction in utility.

Now that we identified how utility information is used toward finding the fair utility for a network link, one question arises: how the individual values from all flows are aggregated into one value? The utility function information at the flow operating utility can be sent in each packet, but would simple addition of this information from all packets into one

aggregate value result in the correct one? It will certainly not. To reach a correct aggregate value, only one value from each flow needs to be added. To overcome this problem, Utilite uses the concept of information splitting. One value, A , can be delivered from a source to a destination over a time period T using L packets if A is split over the L packets. In other words, A can be transmitted using L packets during a time period T if each packet carries A/L , which is added together at the destination over T . We will show in the next section how information splitting is deployed in Utilite.

Although information splitting demonstrates scalability in information delivery, it is vulnerable to packet loss. Since information is distributed over packets spanning the period T , dropping a packet or more results in losing part of the carried information. To overcome this problem, Utilite uses an adjustment weight to correct the information in delivery in case a packet drop occurs. We mentioned earlier that, during congestion, packets are accepted with a probability P_a . This probability can be used as a weighting factor to the information pieces in the accepted packets. By dividing the information parts in accepted packets by the acceptance probability, all values in the packets will be scaled up to compensate for the information in the lost packet.

6.4.4 Overhead Evaluation

Achieving utility-based rate allocation in a scalable fashion involves transporting utility curve information from the sources to the network routers. Being independent of the measurable flow and network parameters, this information has to be delivered from the sources. Consequently, the overhead involved in each packet in Utilite will naturally become larger than the overhead of rate-based allocation mechanisms because the latter is a function of only measurable parameters such as the flow rate and aggregate rate. The parameters that are transported in Utilite header of each packet are listed in Table 1 along with their description. Some of these parameters are shown in Figure 43. Most of the header parameters are used to approximate the utility functions. The header size constitutes less than 20 bytes. If each packet size is 1kB, then the overhead of this header will not exceed 2%, which is reasonable.

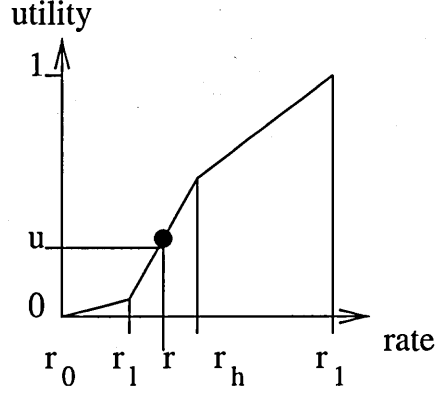


Figure 43: Utility function approximation showing some of the parameters carried in the Utilite header.

Table 1: Utilite parameters and description.

| Value | Description |
|-------|--------------------------------|
| r | see Figure 43 |
| r_l | |
| r_h | |
| r_0 | |
| r_1 | |
| s_i | U_i slope at operating point |
| w | Packet loss compensation |

The major part of the overhead is to deliver the utility function description information. This additional information is needed to maintain a reasonable level of utility function accuracy and algorithm stability.

6.5 Utilite Design

In order to achieve utility fairness between flows in a congested network link, two conditions have to be satisfied: (1) the flow utility, u_i , does not exceed the fair utility, u_f , and (2) the sum of the rates of all flows in a link is equal to its capacity, C , as given in equation (43).

$$u_i \leq u_f, \forall i \in \{1, 2, \dots, N\}; \quad \sum_{i=1}^N r_i = C \quad (43)$$

When congestion occurs, all flows with $u_i > u_f$ will encounter packet loss enough to

bring their utility down to u_f , resulting in a corresponding aggregate rate equal to the link capacity.

Once u_f is derived, the flow corresponding rate, $r_i(u_f)$, can be calculated using the utility function information carried in the packet DPS, then the dropping probability for each flow can be easily computed using the current flow rate as given in equation (42). Therefore, we start by presenting how the proposed algorithm approximates u_f and then calculates its corresponding fair rate.

Since congestion is detected when the aggregate rate exceeds the link capacity, the aggregate rate is used as a tool to adjust the utility at the link to achieve the fair utility as define in equation (43). We define F as the rate accepted by Utilite using the current utility value. The network router measures F to determine whether the current utility value allows accepted rate higher or lower than C . If $F > C$, the excess aggregate traffic is used along with utility functions information to approximate the amount by which the current utility should be decreased. Therefore, we update u_f as given in equation (44), where u_f^n and u_f^o are the new and the old link fair utilite, respectively.

$$u_f^n = u_f^o - (1 - \frac{C}{F}) \times s \quad (44)$$

When $F > C$, u_f^o is updated by subtracting a value proportional to the excess rate in the link, $F - C$. While F is measured at the network router, s is maintained at the network router and calculated using information sent the packet's DPS. s is calculated as given in equation (45), where s_i is the utility function slope for flow i at its operating utility. Notice that s is calculated if $1/s_i$ is delivered using information splitting then aggregated to form $\sum_{i=0}^{N-1} \frac{1}{s_i}$.

$$s = \frac{1}{\sum_{i=0}^{N-1} \frac{1}{s_i}} \quad (45)$$

Once u_f is calculated, the corresponding fair rate can be found using the utility function information carried in the packet. Then, the acceptance probability p_a can be calculated as given in equation (46). Notice that $r_i(u_f^o)$ and $r_i(u_f^n)$ are computed using utility function

information for application i delivered to the router. Notice also that only two values are maintained at the router: s and F . Since no additional individual flow-specific information is maintained, this algorithm is scalable.

$$p_a = \frac{r_i(u_f^o) - r_i(u_f^n)}{r_i(u_f^o)} \quad (46)$$

6.6 Simulation and Results

6.6.1 Simulation Environment

The proposed utility architecture is evaluated using the network simulator NS-2. The network topology, shown in Figure 44, consists of ten traffic sources sharing a congested link. The core network is composed of three routers between the sources and destinations. Each source node generates CBR traffic with all packets from a flow traverse the same path.

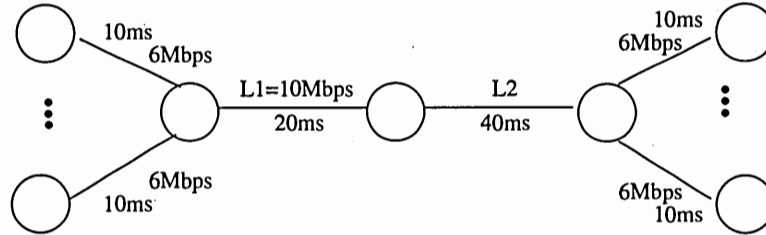


Figure 44: Network topology.

The sources adapt their rate in response to feedback from the destination nodes. The feedback message contains the minimum rate that can be provided by all the routers on the flow's path given the current network condition. We use ten different utility functions that the source nodes can choose from. The ten utility functions are shown in Figure 45: Two are with multiple segments while the other eight are simply straight lines with different slopes.

6.6.2 Results

Different scenarios are used to test the proposed architecture. These scenarios are intended to demonstrate the behavior of Utilite under several network situations. Scenarios are

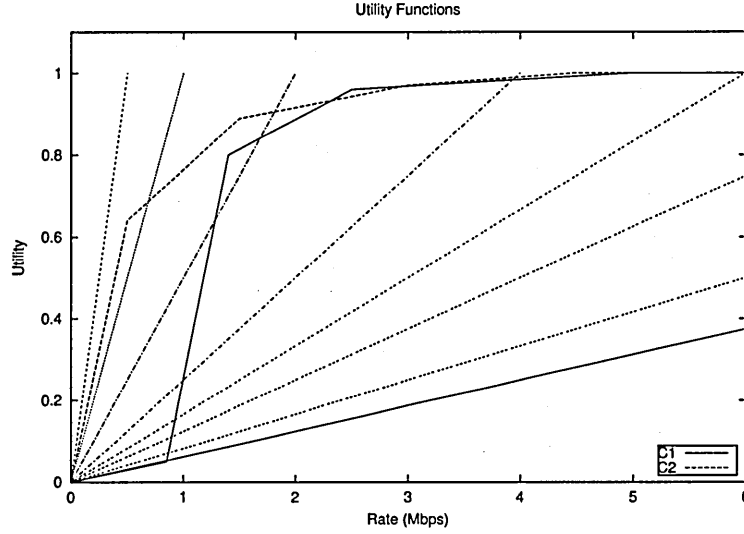


Figure 45: Utility functions used in simulation.

generated by changing some simulation parameters such as the number flows in the link, utility function types used, number of congested links, etc. These scenarios are:

- **One congested link with homogeneous flows:** In this case, all flows have the same utility function and are traversing the same congested link. This case is simple and only shows how Utilite achieves the fair utility for all flows.
- **One congested link with heterogeneous flows:** In this case, we change the number of utility functions used by the ten flows. This scenario shows Utilite scalability with respect to the number of different utility functions it manages. It also shows that flows with different utility functions are allocated different rates.
- **Two consecutive congested links with heterogeneous flows:** This case represents the robustness of Utilite with packet loss. In this case, heterogeneous flows traverse two consecutive links (L_1 and L_2 shown in Figure 44), where L_2 has less capacity than L_1 . L_1 will drop packets from flows until the sources adapt to its link capacity. At the same time, L_2 will also drop packets because its capacity is smaller than L_1 . Eventually, the source nodes should adapt their rates according to the link with the smaller capacity (L_2).

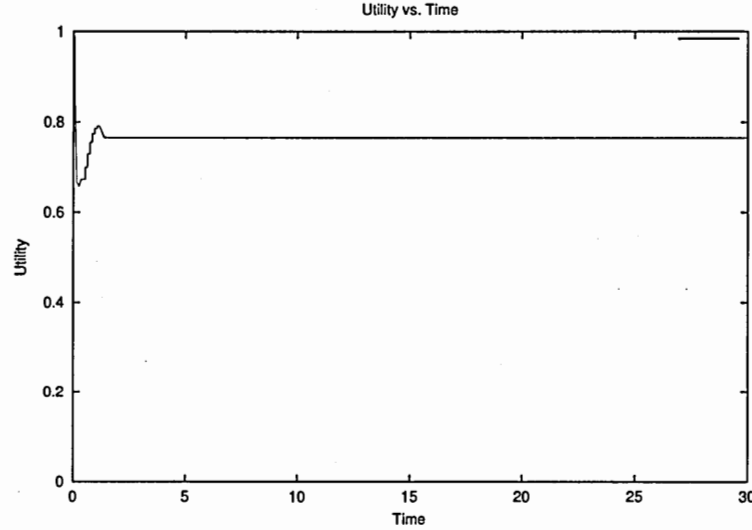


Figure 46: Fair utility for flow with the same utility function.

- **One congested link with some flows with constrained rate:** Some flows may have constrained rate beyond which they cannot increase like when their rate is limited by other congested links in the flows route. We mainly evaluate the case when the fair utility provided by a certain link exceeds the flow's selected operating utility.
- **Dynamic flows entering and leaving a link:** In this case, we consider several heterogeneous flows joining and leaving the network link and evaluate the efficiency of Utilite. We show how flows sharing the link will dynamically increase or decrease their rate according the number of flows and the available bandwidth at the link.

We first consider the scenario of homogeneous flows sharing a link. Each flow has an initial rate of 4Mbps with utility function C1 while the shared link capacity is 10Mbps. Figures 46 and 47 show the fair link utility versus time and the aggregate and individual rates accepted by Utilite versus time, respectively. It is shown that the fair link utility converges to the max-min utility of 0.765. It is also shown that all homogeneous flows are allocated the same bandwidth while the aggregate bandwidth is equal to the link capacity.

Now we consider heterogeneous flows sharing a link. Three cases are considered for the number of utility functions used by the flows: two, five, and ten, which are shown in Figure

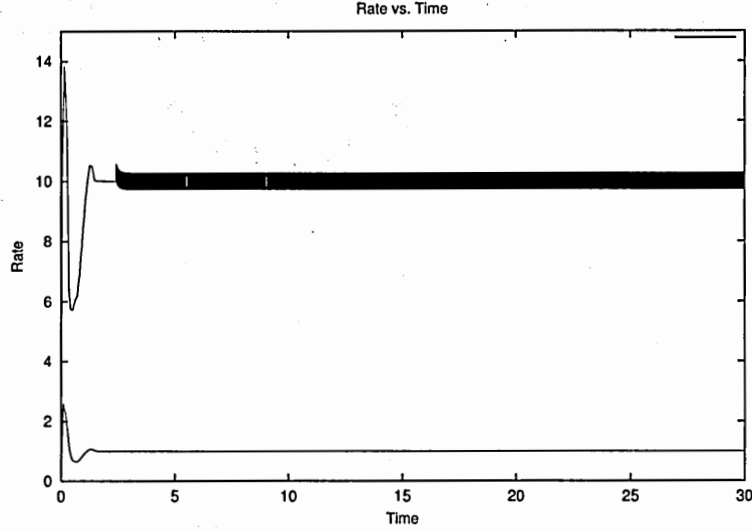
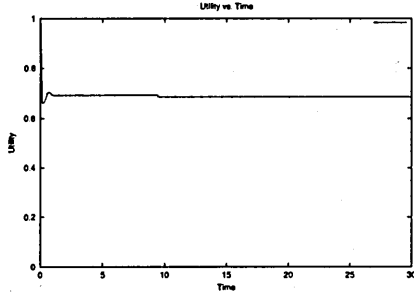


Figure 47: Rate for flows with the same utility functions.

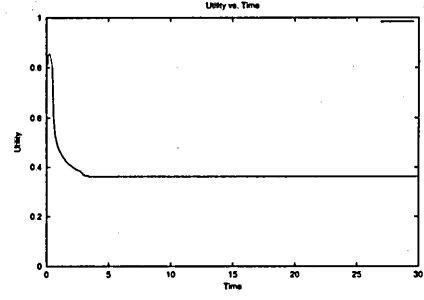
48. It is shown in 48(d) that flow rates are separated into two bands because these flows have one of two utility functions. When the number of different utility functions increase (five or ten), the number of separated rate bands increases as well, as shown in Figures 48 (e) and (f). This shows that Utilite is scalable with the number of different utility functions and correctly allocates rates to flows according to their utility functions.

In the third scenario, we consider two congested links, the capacity for L_1 is 10Mbps and the capacity for L_2 is 9Mbps. Figures 49 and 50 show the utility for L_1 and L_2 , respectively. Also, Figure 51 shows the aggregate and individual accepted rate for L_2 . As shown in the figures, L_2 will be the effective congested link eventually when the fair utility for L_2 becomes lower than that for L_1 . Notice that the utility for L_1 increases to one because it becomes not congested.

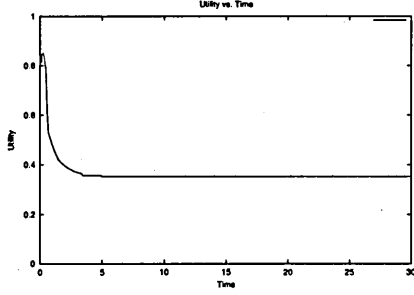
In the fourth scenario, we repeat the first scenario but with some flows that have limited maximum rate. We set the maximum rate for five flows to 0.9Mbps while the other five are unconstrained. Figures 52 and 53 show the fair utility and the aggregate and individual rate for L_1 respectively. We notice that the fair utility of the link is higher than in the first scenario because the unconstrained flows increase their rates to fully utilize the link bandwidth, which results in forming two rate bands (the higher band is for the unconstrained



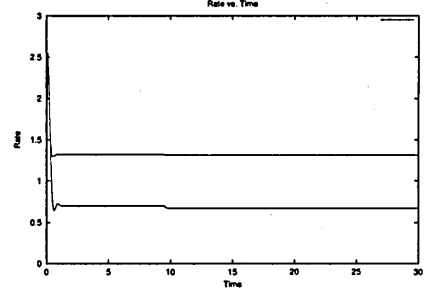
(a) Two utility functions



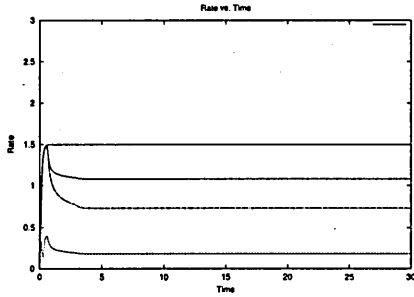
(b) Five utility functions.



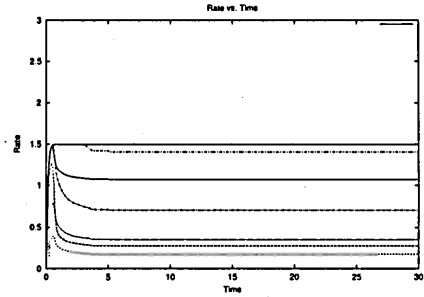
(c) Ten utility functions.



(d) Two utility functions.



(e) Five utility functions.



(f) Ten utility functions.

Figure 48: Utilite scalability with the number of different utility functions. Ten flows are used in all cases but the number of different utility functions varies. In figures:(d),(e), and (f), aggregate link rate and individual flow rates are shown.

flows).

In the last scenario, we consider a dynamic network condition where the number of flows in the link dynamically changes as shown in Figure 54. Figures 55 and 56 show the aggregate and individual accepted rate and the fair utility for L_1 , respectively. We notice that the fair utility changes dynamically to adjust the flows rate in order to utilize the link bandwidth. Three different utility functions are used in this case. It is noticed here again the individual rate separation into three bands, most clearly shown between 10 and 25sec.

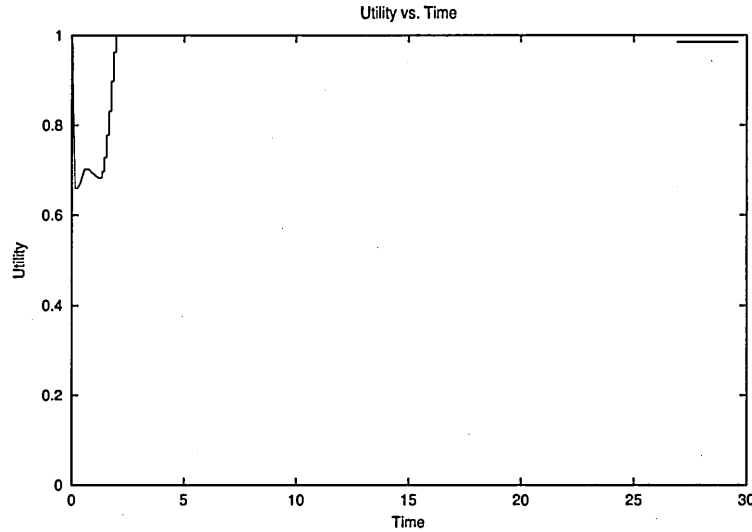


Figure 49: Fair utility at the first congested link with two different utility functions.

6.7 Issues and Discussion

As mentioned earlier, the pricing model for utility-based fair rate allocation is different than that for rate-based allocation due to the difference among flows in their network resources usage. Routers implementing utility-based fair rate allocation assign bandwidth according to the flow utility. If a flat rate pricing model is deployed, however, greedy sources will try to get more bandwidth at no extra cost by cheating and adjusting their utility functions. To prevent greedy sources from dominating the link bandwidth, the pricing model has to be based on the consumed bandwidth by each flow. Therefore, the more bandwidth a source uses, the more it has to pay. Being bandwidth dependent, the pricing model will make the users' budget a limiting factor to the bandwidth they request. Although this bandwidth-based pricing limit the maximum requested bandwidth, it does not efficiently prevent sources from greedily express utility function shapes that can always achieve the maximum requested bandwidth. This can be done by expressing low utility for any bandwidth value below the maximum affordable bandwidth. For example, sources can eventually express a utility function that is more or less a step function with the step edge at the maximum affordable bandwidth. To discourage such behavior, the pricing model

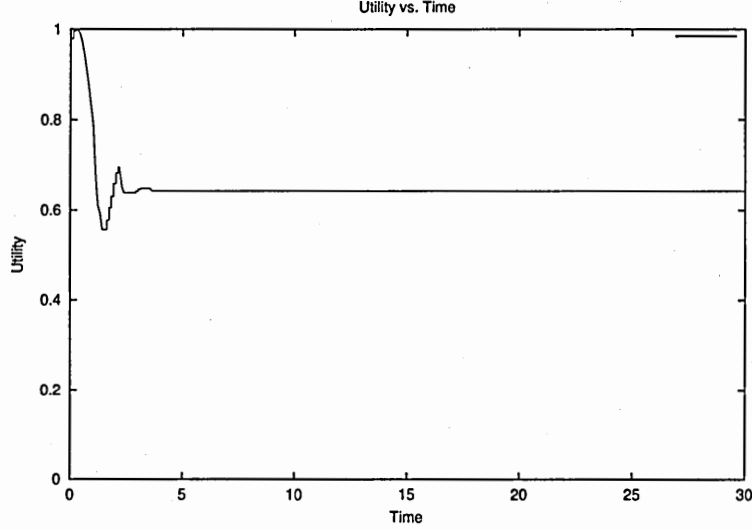


Figure 50: Fair utility at the second congested link with two different utility functions.

should discourage sources from expressing utilities lower than their actual utility at any given bandwidth. In other words, a source will pay less for a given bandwidth usage when its utility function indicates higher utility.

Based on the above discussion, we discuss pricing at two levels: system level and customer level. The system level pricing model is transparent to the user and computes prices according to the usage of network resources. The customer level pricing model, however, produces a fixed price that the customer needs to pay for using the network by taking into account the required service. We propose a skeleton for a pricing model for each of the levels discussed above.

At the system level, we propose a pricing model for utility-based rate allocation that is based on the consumed bandwidth and the link fair utility. The price per flow increases with the consumed bandwidth and decreases with link utility. Assuming a congested link with fair link utility less than one, the price for flow i , p_i , is given in equation (47), where P is the link price, C is the link capacity, N is the number of flows, u_f is the link fair utility, and α and β are constants.

$$p_i = \frac{P(\alpha B - \beta u_f)}{\alpha C - N\beta u_f} \quad (47)$$

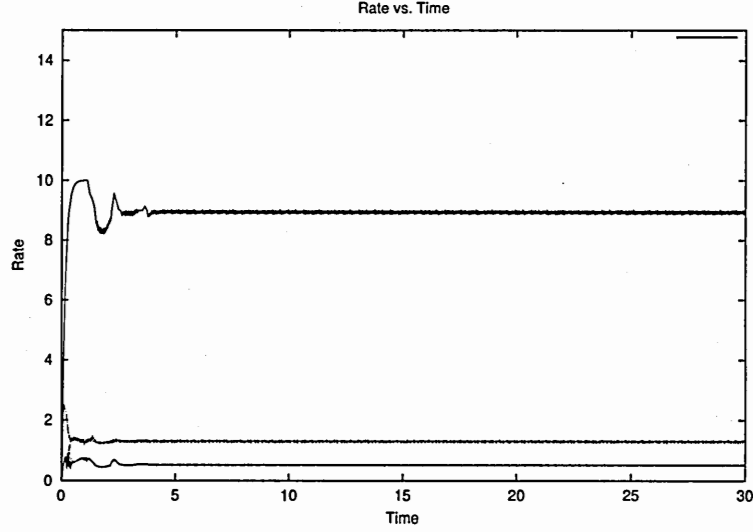


Figure 51: Rate at the second congested link with two different utility functions.

Notice that the vendor will have constant price for the link and will not gain additional profit by allowing more flows to share the link. Instead, the more flows accepted in the link at the expense of lowering the link utility, the lower the price each flow has to pay.

At the consumer level, we propose a pricing model that is also utility based and accounts for the consumed bandwidth. This model produces a fixed price for each user depending on the shape of their utility function. The price, p_i , for user i with a utility function $U_i(r)$ is in the form given in equation (48) where α is a constant.

$$p_i = \alpha \int_{r=0}^{\infty} x(1 - U(x))dx. \quad (48)$$

Notice that p_i increases when the user's utility function indicates low utility (or low satisfaction level) at higher bandwidth value, which decreases the tendency of a user to cheat when submitting their utility functions to the network.

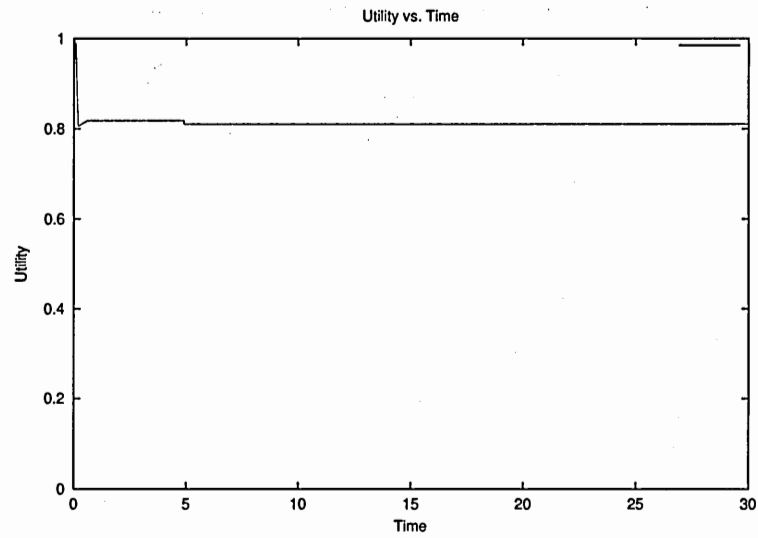


Figure 52: Fair utility change with some flows with limited rate.

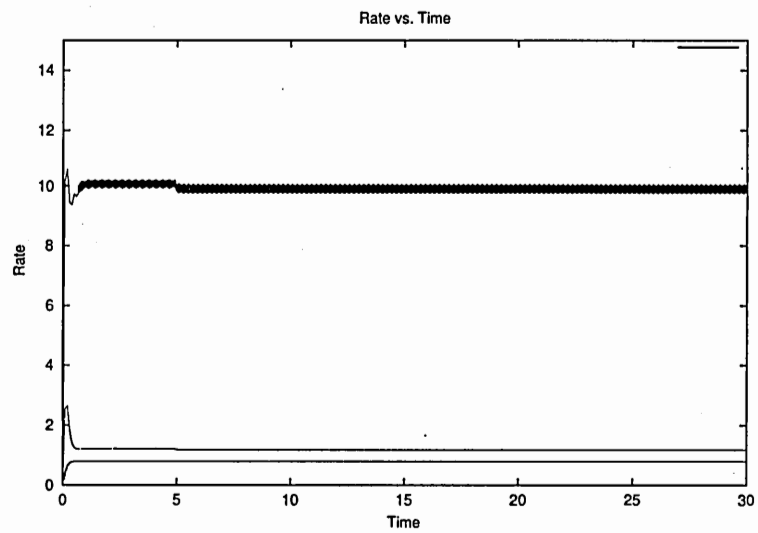


Figure 53: Rate change with some flows with limited rate.

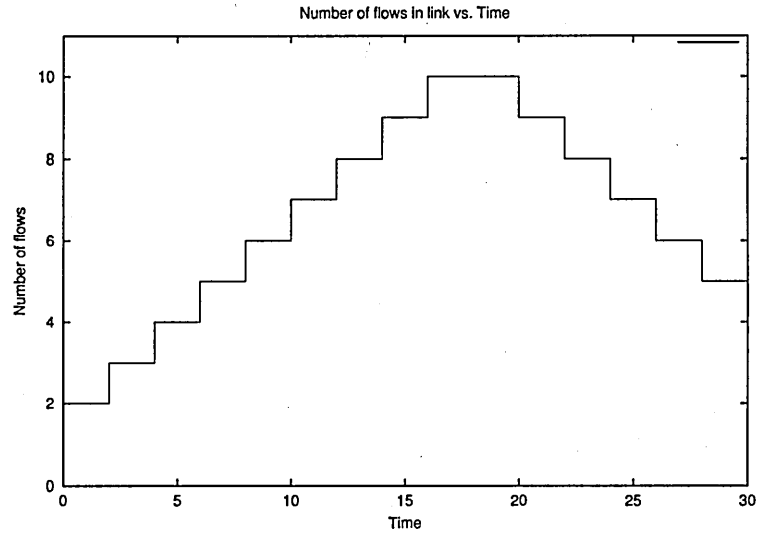


Figure 54: Rate Change for dynamic flows.

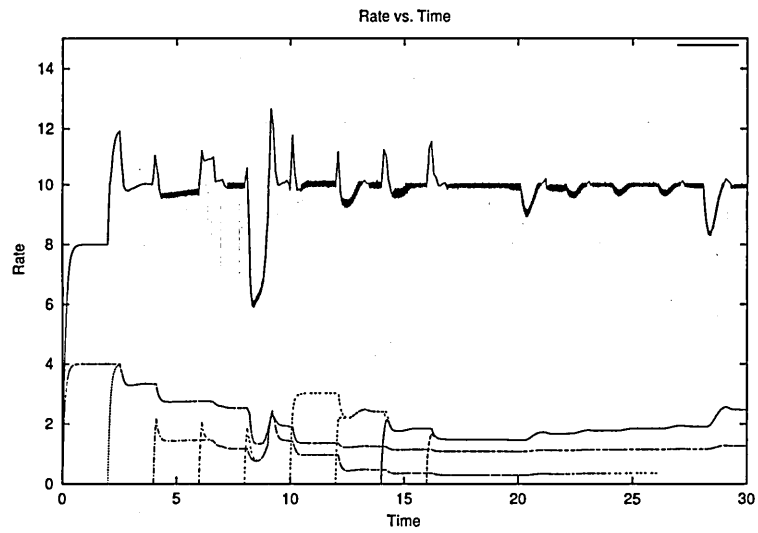


Figure 55: Rate Change for dynamic flows.

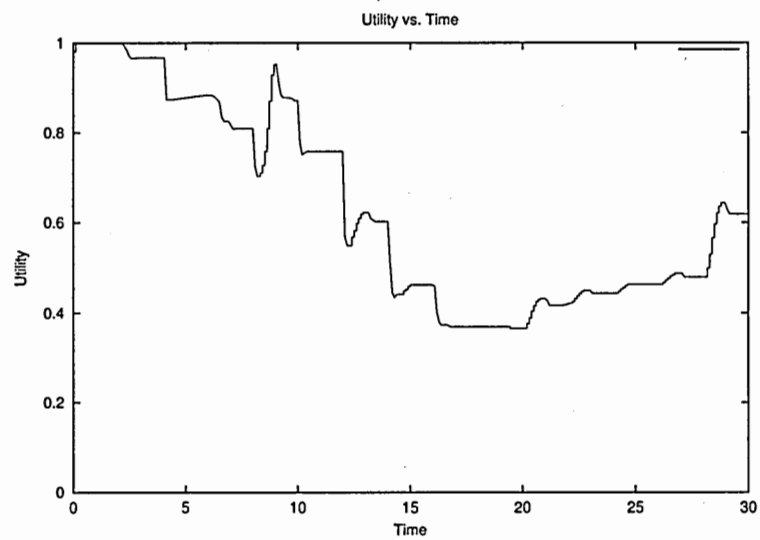


Figure 56: Fair utility change for dynamic flows.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

In this thesis, we studied several router mechanisms and proposed scalable application-aware solutions that enhance their performance. The router mechanisms studies are buffer management and rate allocation.

We started by evaluating the performance of several existing buffer management schemes using goodput as the metric. We also proposed a goodput-based estimation methodology using Markov chains to estimate the performance of the buffer management schemes.

After evaluating the performance of these buffer management schemes, we proposed a new simple buffer management scheme, MPFD, that is application-aware and achieves the performance of complex buffer management schemes. MPFD is evaluated using MPEG-2 traffic as an example, but it can be used with other applications that utilize application-aware buffer management schemes.

In rate allocation, we proposed Utilite: a new scalable utility-based fair rate allocation architecture. Utilite leverages utility functions for all flows sharing a network link. The fairness criteria applied in this work is max-min fairness. The architecture achieved fairness for all flows in the network in a scalable manner. The simulation results demonstrated the correctness and accuracy of Utilite for several network conditions. It also shown how flows with different utility functions are allocated different rates while maintaining fair utility for all flows.

For future work, we are interested in evaluating the performance of a network node that is using both MPFD and Utilite. We are also interested in looking on other techniques to support an application-aware network environment such as network protocols.

APPENDIX A

TBS QUEUEING ANALYSIS

We present the C_a matrices of TBS scheme. Following the discussions in Section 4.3.2 and Subsection 4.6.1, C_a can be easily constructed. The state order for the matrices in this section is shown below.

- For I-frames: *accept*, *drop GOPD*, *drop GOPA*, *drop frame**, *drop GOPD**, and *drop GOPA**.
- For P-frames: *accept*, *drop GOPD*, *drop GOPA*, and *drop frame**.
- For B-frames except the last two: *accept*, *drop GOPD*, *drop GOPA*, *drop frameD*, and *drop frameA*.
- For the last two B-frames: *accept*, *drop GOPD*, *drop GOPA*, *drop frameD*, *drop frameA*, *drop GOPD**, and *drop GOPA**.

The C_a transition matrices for I-frame and the transition from the last slot of the I-frame to the first slot of the following B-frame are given in equations (49) and (50), respectively.

$$C_a = \begin{bmatrix} D_{f_B} & D_{t_B} & 0 & 0 & 0 \\ 0 & D_{c_B} & 0 & 0 & 0 \\ 0 & D_{t_B} & D_{f_B} & 0 & 0 \\ 0 & D_{t_B} & 0 & D_{f_B} & 0 \\ 0 & D_{t_B} & 0 & 0 & D_{f_B} \end{bmatrix} \quad (49)$$

$$C_a = \begin{bmatrix} D_{f_H} & 0 & 0 & D_{t_H} & 0 & 0 & 0 \\ 0 & D_{t_L} & D_{f_L} & 0 & 0 & 0 & 0 \\ D_{f_L} & 0 & 0 & D_{t_L} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & D_{t_L} & D_{f_L} \\ 0 & 0 & 0 & 0 & 0 & D_{t_H} & D_{f_H} \end{bmatrix} \quad (50)$$

The C_a transition matrices for P-frame and the transition from the last slot of the I-frame to the first slot of the following B-frame are given in equations (51) and (52), respectively.

$$C_a = \begin{bmatrix} D_{f_B} & D_{t_B} & 0 & 0 \\ 0 & D_{c_B} & 0 & 0 \\ 0 & 0 & D_{c_B} & 0 \\ 0 & D_{t_B} & 0 & D_{f_B} \end{bmatrix} \quad (51)$$

$$C_a = \begin{bmatrix} D_{f_H} & 0 & 0 & D_{t_H} & 0 \\ 0 & D_{t_L} & D_{f_L} & 0 & 0 \\ 0 & D_{t_H} & D_{f_H} & 0 & 0 \\ D_{f_L} & 0 & 0 & D_{t_L} & 0 \end{bmatrix} \quad (52)$$

The C_a transition matrix for B-frames, except the last two frames, is given in equation (53).

$$C_a = \begin{bmatrix} D_{f_H} & 0 & 0 & D_{t_H} & 0 \\ 0 & D_{t_L} & D_{f_L} & 0 & 0 \\ 0 & D_{t_H} & D_{f_H} & 0 & 0 \\ 0 & 0 & 0 & D_{t_L} & D_{f_L} \\ 0 & 0 & 0 & D_{t_H} & D_{f_H} \end{bmatrix} \quad (53)$$

The C_a transition matrix from last slot of a B-frame, except the last two frames, to the first slot of the B-frame, I-frame, and P-frame, are given in equations (54), (55), and (56), respectively.

$$C_a = \begin{bmatrix} D_{f_H} & 0 & 0 & D_{t_H} & 0 \\ 0 & D_{t_L} & D_{f_L} & 0 & 0 \\ 0 & D_{t_H} & D_{f_H} & 0 & 0 \\ D_{f_L} & 0 & 0 & D_{t_L} & 0 \\ D_{f_H} & 0 & 0 & D_{t_H} & 0 \end{bmatrix} \quad (54)$$

$$C_a = \begin{bmatrix} D_{f_B} & D_{t_B} & 0 & 0 & 0 \\ 0 & D_{t_B} & 0 & D_{f_B} & 0 \\ 0 & D_{t_B} & 0 & 0 & D_{f_B} \\ 0 & D_{t_B} & D_{f_B} & 0 & 0 \\ D_{f_B} & D_{t_B} & 0 & 0 & 0 \end{bmatrix} \quad (55)$$

$$\mathbf{C}_a = \begin{bmatrix} \mathbf{D}_{f_B} & \mathbf{D}_{t_B} & 0 & 0 \\ 0 & \mathbf{D}_{c_B} & 0 & 0 \\ 0 & 0 & \mathbf{D}_{c_B} & 0 \\ 0 & \mathbf{D}_{t_B} & 0 & \mathbf{D}_{f_B} \\ \mathbf{D}_{f_B} & \mathbf{D}_{t_B} & 0 & 0 \end{bmatrix} \quad (56)$$

The \mathbf{C}_a transition matrix for the last two B-frames is given in equation (57).

$$\mathbf{C}_a = \begin{bmatrix} \mathbf{D}_{f_H} & 0 & 0 & \mathbf{D}_{t_H} & 0 & 0 & 0 \\ 0 & \mathbf{D}_{t_L} & \mathbf{D}_{f_L} & 0 & 0 & 0 & 0 \\ 0 & \mathbf{D}_{t_H} & \mathbf{D}_{f_H} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{D}_{t_L} & \mathbf{D}_{f_L} & 0 & 0 \\ 0 & 0 & 0 & \mathbf{D}_{t_H} & \mathbf{D}_{f_H} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{D}_{t_L} & \mathbf{D}_{f_L} \\ 0 & 0 & 0 & 0 & 0 & \mathbf{D}_{t_H} & \mathbf{D}_{f_H} \end{bmatrix} \quad (57)$$

For the last two B-frames, the \mathbf{C}_a transition matrices from last slot to the first slot of the following B-frame and P-frame, are given in equations (58) and (59), respectively.

$$\mathbf{C}_a = \begin{bmatrix} \mathbf{D}_{f_H} & 0 & 0 & \mathbf{D}_{t_H} & 0 & 0 & 0 \\ 0 & \mathbf{D}_{t_L} & \mathbf{D}_{f_L} & 0 & 0 & 0 & 0 \\ 0 & \mathbf{D}_{t_H} & \mathbf{D}_{f_H} & 0 & 0 & 0 & 0 \\ \mathbf{D}_{f_L} & 0 & 0 & \mathbf{D}_{t_L} & 0 & 0 & 0 \\ \mathbf{D}_{f_H} & 0 & 0 & \mathbf{D}_{t_H} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{D}_{t_L} & \mathbf{D}_{f_L} \\ 0 & 0 & 0 & 0 & 0 & \mathbf{D}_{t_H} & \mathbf{D}_{f_H} \end{bmatrix} \quad (58)$$

$$\mathbf{C}_a = \begin{bmatrix} \mathbf{D}_{f_B} & \mathbf{D}_{t_B} & 0 & 0 \\ 0 & \mathbf{D}_{c_B} & 0 & 0 \\ 0 & 0 & \mathbf{D}_{c_B} & 0 \\ 0 & \mathbf{D}_{t_B} & 0 & \mathbf{D}_{f_B} \\ \mathbf{D}_{f_B} & \mathbf{D}_{t_B} & 0 & 0 \\ 0 & \mathbf{D}_{t_B} & 0 & \mathbf{D}_{f_B} \\ \mathbf{D}_{f_B} & \mathbf{D}_{t_B} & 0 & 0 \end{bmatrix} \quad (59)$$

APPENDIX B

TBS-D QUEUEING ANALYSIS

The C_a transition matrices for I-frame and the transition from the last slot of the I-frame to the first slot of the following B-frame are given in equations (60) and (61), respectively.

$$C_a = C(t, 0, 1) = \begin{bmatrix} D_f & D_t & 0 & 0 & 0 \\ 0 & D_e & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & D_t & 0 & D_f & 0 \\ 0 & D_t & 0 & 0 & D_f \end{bmatrix} \quad (60)$$

$$C_a = C(T - 1, 0, 1) = \begin{bmatrix} D_{fH} & 0 & D_{tH} & 0 & 0 \\ 0 & D_e & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ D_{fL} & 0 & D_{tL} & 0 & 0 \\ 0 & 0 & 0 & 0 & D_e \end{bmatrix} \quad (61)$$

The C_a transition matrices for P-frame and the transition from the last slot of the I-frame to the first slot of the following B-frame are given in equations (62) and (63), respectively.

$$C_a = C(t, 0, 1) = \begin{bmatrix} D_f & D_t & 0 & 0 \\ 0 & D_e & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & D_t & 0 & D_f \end{bmatrix} \quad (62)$$

$$C_a = C(T - 1, 0, 1) = \begin{bmatrix} D_{fH} & 0 & D_{tH} & 0 \\ 0 & D_e & 0 & 0 \\ 0 & 0 & 0 & 0 \\ D_{fL} & 0 & D_{tL} & 0 \end{bmatrix} \quad (63)$$

The C_a transition matrix for B-frames, except the last two frames, is given in equation (64).

$$\mathbf{C}_a = \mathbf{C}(t, w, 1) = \begin{bmatrix} \mathbf{D}_{f_H} & 0 & \mathbf{D}_{t_H} & 0 & 0 \\ 0 & \mathbf{D}_e & 0 & 0 & 0 \\ 0 & 0 & \mathbf{D}_e & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (64)$$

The \mathbf{C}_a transition matrix from last slot of a B-frame, except the last two frames, to the first slot of the B-frame, I-frame, and P-frame, are given in equations (65), (66), and (67), respectively.

$$\mathbf{C}_a = \mathbf{C}(T-1, w, 1) = \begin{bmatrix} \mathbf{D}_{f_H} & 0 & \mathbf{D}_{t_H} & 0 & 0 \\ 0 & \mathbf{D}_e & 0 & 0 & 0 \\ \mathbf{D}_{f_L} & 0 & \mathbf{D}_{t_L} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (65)$$

$$\mathbf{C}_a = \mathbf{C}(T-1, w, 1) = \begin{bmatrix} \mathbf{D}_f & \mathbf{D}_t & 0 & 0 & 0 \\ 0 & \mathbf{D}_t & 0 & 0 & \mathbf{D}_f \\ \mathbf{D}_{f_L} & \mathbf{D}_t & 0 & \mathbf{D}_f - \mathbf{D}_{f_L} & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (66)$$

$$\mathbf{C}_a = \mathbf{C}(T-1, w, 1) = \begin{bmatrix} \mathbf{D}_f & \mathbf{D}_t & 0 & 0 & 0 \\ 0 & \mathbf{D}_e & 0 & 0 & 0 \\ \mathbf{D}_{f_L} & \mathbf{D}_t & 0 & \mathbf{D}_f - \mathbf{D}_{f_L} & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (67)$$

The \mathbf{C}_a transition matrix for the last two B-frames is given in equation (68).

$$\mathbf{C}_a = \mathbf{C}(t, w, 1) = \begin{bmatrix} \mathbf{D}_{f_H} & 0 & \mathbf{D}_{t_H} & 0 & 0 \\ 0 & \mathbf{D}_e & 0 & 0 & 0 \\ 0 & 0 & \mathbf{D}_e & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{D}_e \end{bmatrix} \quad (68)$$

For the last two B-frames, the \mathbf{C}_a transition matrices from last slot to the first slot of the following B-frame and P-frame, are given in equations (69) and (70), respectively.

$$\mathbf{C}_a = \mathbf{C}(T-1, 1, 1) = \begin{bmatrix} \mathbf{D}_{f_H} & 0 & \mathbf{D}_{t_H} & 0 & 0 \\ 0 & \mathbf{D}_e & 0 & 0 & 0 \\ \mathbf{D}_{f_L} & 0 & \mathbf{D}_{t_L} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{D}_e \end{bmatrix} \quad (69)$$

$$\mathbf{C}_a = \mathbf{C}(T-1, 2, 1) = \begin{bmatrix} \mathbf{D}_f & \mathbf{D}_t & 0 & 0 & 0 \\ 0 & \mathbf{D}_e & 0 & 0 & 0 \\ \mathbf{D}_{f_L} & \mathbf{D}_t & 0 & \mathbf{D}_f - \mathbf{D}_{f_L} & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \mathbf{D}_f & \mathbf{D}_t & 0 & 0 & 0 \end{bmatrix} \quad (70)$$

REFERENCES

- [1] ADAS, A., "Traffic Models in Broadband Networks," *IEEE Communication Magazine*, vol. 35, pp. 82-89, July 1997.
- [2] ALTMAN, E., GALTIER, J., and TOUATI, C., "Utility Based Fair Bandwidth Allocation," *Proceedings of the IASTED International Conference on Networks, Parallel and Distributed Processing and Applications*, October 2002.
- [3] AWAD, A., MCKINNON, M., and SIVAKUMAR, R., "Estimation of Video Queueing Performance Using Markov Chains," *IEEE Global Communications Conference (GLOBECOM)*, Taipei, Taiwan, November 2002.
- [4] AWAD, A., MCKINNON, M., and SIVAKUMAR, R., "Goodput Estimation of an Access Node Buffer Carrying Correlated Video Traffic," *IEEE Symposium on Computers and Communications (ISCC)*, Taormina, Italy, pp. 120-124, July 2002.
- [5] AWAD, A., MCKINNON, M., and SIVAKUMAR, R., "MPFD: A Lookahead Based Buffer Management Scheme for MPEG-2 Video Traffic," *IEEE Symposium on Computers and Communications (ISCC)*, Kemer - Antalya, Turkey, June 2003.
- [6] BERNET, Y., FORD, P., YAVATKAR, R., BAKER, F., ZHANG, L., SPEER, M., BRADEN, R., DAVIE, B., WROCLAWSKI, J., and FELSTAIN, E., "A Framework for Integrated Services Operation over Diffserv Networks," *IETF Network Working Group*, <http://www.ietf.org/rfc/rfc2998.txt>, November 2000.
- [7] BERTSEKAS, D. and GALLAGER, R., *Data Networks*, vol. chapter 6. 1987.
- [8] BLAKE, S., BLACK, D., CARLSON, M., E., D., WANG, Z., and WEISS, W., "An Architecture for Differentiated Services," *IETF Network Working Group*, <http://www.ietf.org/rfc/rfc2475.txt>, December 1998.
- [9] BOX, G. E. P. and JENKINS, G. M., *Time Series Analysis: Forecasting and Control*. Holden-Day series in time series analysis., San Francisco, Holden-Day, 1970.
- [10] BRADEN, R., ZHANG, L., BERSON, S., HERZOG, S., and JAMIN, S., "Resource Reservation Protocol (RSVP)- Version 1 Functional Specification," *IETF Network Working Group*, <http://www.ietf.org/rfc/rfc2205.txt>, September 1997.
- [11] BRAGG, A. and CHOU, W., "Analytic Models and Characteristics of Video Traffic in High Speed Networks," *MASCOTS '94. Proceedings of the Second International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, pp. 68-73, Jan. 1994.
- [12] CALVERT, K., BHATTACHARJEE, S., ZEGURA, E., and STERBENZ, J., "Directions in Active Networks," *IEEE Communications Magazine*, vol. 36, pp. 72-8, October 1998.
- [13] CAO, Z. and ZEGURA, E., "Utility Max-Min: An Application-Oriented Bandwidth Allocation Scheme," *INFOCOM*, vol. 2, pp. 793-801, March 1999.

- [14] CAO, Z., WANG, Z., and ZEGURA, E., "Rainbow Fair Queueing: Fair Bandwidth Sharing Without Per-Flow State," *Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 2, pp. 922–31, March 2000.
- [15] CHATFIELD, C., *The Analysis of Time Series : Theory and Practice*. Monographs on applied probability and statistics., London : Chapman and Hall ; New York : Wiley., 1975.
- [16] CHIRUVOLU, G., DAS, T. K., SANKAR, R., and RANGANATHAN, N., "A Scene-based Generalized Markov Chain Model for VBR Video Traffic," in *IEEE International Conference on Communications*, pp. 554–8, June 1998.
- [17] CUENCA, P., GARRIDO, A., QUILES, F., and OROZCO-BARBOSA, L., "Performance Evaluation of Cell Discarding Mechanisms for the Distribution of VBR MPEG-2 Video Over ATM Networks," *IEEE Transactions on Broadcasting*, vol. 44, June 1998.
- [18] FENG, I., LO, K.-T., MEHRPOUR, H., and KARBOWIAK, A., "Cell Loss Concealment Method for MPEG Video in ATM Networks," *IEEE Global Telecomm. Conference.*, vol. 3, pp. 1925–9, Nov. 1995.
- [19] FREY, M. and NGUYEN-QUANG, S., "A Gamma-Based Framework for Modeling Variable MPEG Video Sources: The GOP GBAR Model," *IEEE/ACM ToN*, vol. 8, pp. 710–9, December 2000.
- [20] FROSSARD, P. and VERSCHEURE, O., "AMISP: A Complete Content-Based MPEG-2 Error-Resilient Scheme," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, pp. 989–98, Sept. 2001.
- [21] FROST, V. S. and MELAMED, B., "Traffic Modeling For Telecommunications Networks," *IEEE Communication Magazine*, pp. 70–81, 1993.
- [22] GAO, X., NANDAGOPAL, T., and BHARGHAVAN, V., "Achieving Application Level Fairness Through Utility-Based Wireless Fair Scheduling," *IEEE Global Telecommunications Conference, 2001. GLOBECOM 01*, vol. 6, pp. 3257–61, Nov. 2001.
- [23] GARRETT, M. W. and WILLINGER, W., "Analysis, Modeling and Generation of Self-Similar VBR Video traffic," in *SIGCOMM '94*, (London, UK), pp. 269–80, Sept. 1994.
- [24] HAN, T. and OROZCO-BARBOSA, L., "Performance Requirements for the Transport of MPEG Video Streams over ATM Networks," *IEEE International Conference on Communications*, vol. 1, pp. 221–5, June 1995.
- [25] HEDRICK, C., "Routing Information Protocol," *IETF*, <http://www.ietf.org/rfc/rfc1058.txt>, 1988.
- [26] JACOBSON, V., NICHOLS, K., and PODURI, K., "Receiver-Driven Layered Multicast," *Proceedings of ACM SIGCOMM*, October 1996.
- [27] JANG, B. S. and THOMSON, C., "Threshold Autoregressive Models for VBR MPEG Video Traces," in *INFOCOM '98*, (San Francisco, California), p. 209, March/April 1998.

- [28] JIANXIAN, L., LING, Z., FENG SUILI LIN, J., and YOSHIDA, J., "A Survey of RSVP Performance," *Fifth Asia-Pacific Conference on Communications and Fourth Optoelectronics and Communications Conference. APCC/OECC'99.*, vol. 2, pp. 1200-3, October 1999.
- [29] JIAOYANG, L., HAIFENG, X., and XIANDE, L., "The Importance of Multiple Correlation of Variable Bit Rate Video Traffic in ATM Traffic Engineering," *Proceedings of the SPIE - The International Society for Optical Engineering*, vol. 3561, pp. 423-9, Sept. 1998.
- [30] KANG, C. and TAN, H., "Queueing Analysis of Explicit Priority Assignment Partial Buffer Sharing Schemes for ATM Networks," *INFOCOM*, vol. 2, pp. 810 -819, April 1993.
- [31] KAR, K., SARKAR, S., and TASSIULAS, L., "A Simple Rate Control Algorithm for Maximizing Total User Utility," *INFOCOM*, vol. 1, pp. 133-41, April 2001.
- [32] KAWAHARA, K., KITAJIMA, K., TAKINE, T., and OIE, Y., "Packet Loss Performance of Selective Cell Discard Schemes in ATM Switches," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 5, pp. 903-13, 1997.
- [33] KELLY, F. P., "Charging and Rate Control for Elastic Traffic," *European Transactions on Telecommunications*, vol. 8, pp. 33-37, January 1997.
- [34] KRUNZ, M. and HUGHES, H., "A Performance Study of Loss Probability for MPEG Video Traffic," *IEEE International Conference on Communications*, vol. 3, pp. 1756-60, 1995.
- [35] KRUNZ, M., SASS, R., and HUGHES, H., "Statistical Characteristics and Multiplexing of MPEG Streams," *INFOCOM'95*, vol. 2, pp. 455-62, April 1995.
- [36] KRUNZ, M. and TRIPATHI, S. K., "Scene-Based Characterization of VBR MPEG-Compressed Video Traffic," Tech. Rep. CS-TR-3573, UMIACS TR-95-120, Department of Computer Science, University of Maryland, College Park, Maryland, 1996.
- [37] LA, R. and ANANTHARAM, V., "Utility-Based Control in the Internet for Elastic Traffic," *IEEE TON*, vol. 10, pp. 272-86, April 2002.
- [38] LI, J. R., GAO, X., QIAN, L., and BHARGHAVAN, V., "Goodput Control for Heterogeneous Data Streams," *NOSSDAV*, June 2000.
- [39] LIN, D. and MORRIS, R., "Dynamics of Random Early Detection," *Proceedings of ACM SIGCOMM*, pp. 127-37, October 1997.
- [40] LOU, W., CHIA, L. T., and LEE, B. S., "Characterization and Source Modeling of MPEG-2 VBR Video Source," in *ICICS '97*, (Singapore), pp. 1652-56, Sept. 1997.
- [41] MITCHELL, J. L., *MPEG Video Compression Standard*. Digital Multimedia Standards Series., New York:Chapman & Hall, 1997.
- [42] MOY, J., "OSPF Version 2," *IETF*, <http://www.ietf.org/rfc/rfc1583.txt>, 1994.

- [43] NICHOLS, K., BLAKE, S., BAKER, F., and BLACK, D., "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers," *IETF Network Working Group*, <http://www.ietf.org/rfc/rfc2474.txt>, December 1998.
- [44] NOMURA, M., FUJII, T., and OHTA, N., "Basic Characteristics of Variable Rate Video Coding in ATM Environments," *IEEE Journal on Selected Areas on Communication*, vol. 7, pp. 752-760, June 1989.
- [45] PAREKH, A. and GALLAGER, R., "A Generalized Processor Sharing Approach to Flow Control," *Proceedings of the INFOCOM*, 1992.
- [46] REININGER, D. and IZMAILOV, R., "Soft Quality-of-Service with VBR+ Video," *In the Proceedings of 8th International Workshop on Packet Video*, vol. 13, pp. 1176-88, September 1997.
- [47] REKHTER, Y. and LI, T., "A Border Gateway Protocol 4 (BGP-4)," *IETF*, <http://www.ietf.org/rfc/rfc1771.txt>, 1995.
- [48] ROSE, O., "Statistical Properties of MPEG Video Traffic and Their Impact on Traffic Modeling in ATM Systems," in *20th Conference on Local Computer Networks*, pp. 397-406, Oct. 1995.
- [49] ROSE, O., "Discrete-time Analysis of a Finite Buffer with VBR MPEG Video Traffic Input," *Proceedings of the 15th International Teletraffic Congress- ITC 15*, vol. 2, pp. 413-22, 1997.
- [50] ROSE, O., "Simple and Efficient Models for Variable Bit Rate MPEG Video Traffic," *Performance Evaluation*, vol. 30, pp. 69-85, July 1997.
- [51] SARKAR, S. and TASSIULAS, L., "Fair Allocation of Utilities in Multirate Multicast Networks: A Framework for Unifying Diverse Fairness Objectives," *IEEE Transactions on Automatic Control*, vol. 47, pp. 931-44, June 2002.
- [52] SHENKER, S., "Fundamental Design Issues for the Future Internet," *JSAC*, vol. 13, pp. 1176-88, September 1995.
- [53] SIVAKUMAR, R., KIM, T., VENKITARAMAN, N., and BHARGHAVAN, V., "Achieving Per-Flow Weighted Rate Fairness in a Core-Stateless Network," *International Conference on Distributed Computing Systems (ICDCS)*, pp. 188-96, April 2000.
- [54] STEWART, W. J., *Introduction to Numerical Solution of Markov Chains*. New Jersey : Princeton, 1994.
- [55] STOICA, I., SHENKER, S., and ZHANG, H., "Core-Stateless Fair Queueing: Achieving Approximately Fair Bandwidth Allocations in High Speed Networks," *SIGCOMM*, vol. 28, pp. 118-30, October 1998.
- [56] STOICA, I. and ZHANG, H., "LIRA: A Model for Service Differentiation in the Internet," *NOSSDAV*, 1999.
- [57] TAN, C. H. and ZHANG, L., "Effects of Cell Loss on the Quality of Service for MPEG Video in ATM Environment," *IEEE Singapore International Conference on Communications and Networks*, pp. 11-15, July 1995.

- [58] TENNENHOUSE, D. L. and WETHERALL, D. J., "Towards An Active Network Architecture," *Multimedia Computing and Networking*, January 1996.
- [59] TIAN, T., LI, A., WEN, J., and VILLASENOR, J., "Priority Dropping in Network Transmission of Scalable Video," *International Conference on Image Processing*, vol. 3, pp. 400-3, Sept. 2000.
- [60] VENKITARAMAN, N., MYSORE, J., and NEEDHAM, M., "A Core-Stateless Utility Based Rate Allocation Framework," *IFIP/IEEE International Workshop Protocols for High Speed Networks, PfHSN*, vol. 2334, pp. 1-16, April 2002.
- [61] VERSCHEURE, O., FROSSARD, P., and HAMDI, M., "Joint Impact of MPEG-2 Encoding Rate and ATM Cell Losses on Video Quality," *GLOBECOM '98*, vol. 1, pp. 71-6, Nov. 1998.
- [62] WHITE, P., "RSVP and Integrated Services in the Internet: a Tutorial," *IEEE Communications Magazine*, vol. 35, pp. 100-6, May 1997.
- [63] WHITE, P. and CROWCROFT, J., "The Integrated Services in the Internet: State of the Art," *Proceedings of the IEEE*, vol. 85, pp. 1934-46, December 1997.
- [64] WILSON, D. and GHANBARI, M., "An Efficient Loss Priority Scheme for MPEG-2 Variable Bit Rate Video for ATM Networks," *Global Telecommunications Conference.*, vol. 3, pp. 1954-58, Nov. 1996.
- [65] YEGENOGLU, F., JABBARI, B., and ZHANG, Y., "Motion Classified Autoregressive Modeling for Variable Bit Rate Video," in *INFOCOM*, (Florence, Italy), pp. - (1C.4), IEEE, May 1992.
- [66] ZHENG, B. and ATIQUZZAMAN, M., "TSFD: Two Stage Frame Dropping for Scalable Video Transmission Over Data Networks," *IEEE Workshop on High Performance Switching and Routing*, pp. 43 - 47, May 2001.
- [67] ZHU, W., HOU, Y., WANG, Y., and ZHANG, Y.-Q., "End-to-end Modeling and Simulation of MPEG-2 Transport Streams over ATM Networks with Jitter," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, pp. 9-12, Feb. 1998.