ABSTRACT

STUCKEY, JEFFREY J. Worst-Case Cell Delay in an ATM Switch using EDF Scheduling. (Under the direction of Professor George N. Rouskas and Professor Ioannis Viniotis.)

Asynchronous Transfer Mode (ATM) is well-suited for carrying real-time traffic. Because real-time traffic must arrive at its destination prior to its playback instant, end-to-end delay is one of its most critical QoS parameters.

We focus our attention on analyzing delays under worst-case conditions. In particular, our interests are in characterizing the tail behavior of cell delay functions in an ATM switch using the EDF scheduling algorithm. We use simulation techniques to observe cell delays and the behavior of EDF scheduling.

In this study we examine four different source models and their effect on the tails of cell delay functions. We observe that cell delays are affected by many factors such as the source model behavior, the degree of multiplexing, the leaky bucket behavior, and the synchronization of cell transmissions.

Our results suggest that cell delays approach their theoretical maximum delay bound during a switch's warm-up period. Further, we conclude that cell delays increase as the degree of multiplexing decreases. We examine two different methods of worst-case leaky bucket behavior in terms of admitting individual cells and bursts of cells. We find that very large cell delays result in both of these worst-case leaky bucket behaviors. Finally, we show that worst-case cell delays occur when sources are aligned to begin transmitting cells simultaneously and that these delays can be greatly reduced if sources begin transmitting cells at random points in time.

WORST-CASE CELL DELAY IN AN ATM SWITCH USING EDF SCHEDULING

by

Jeffrey J. Stuckey

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Computer Science

Raleigh

1998

APPROVED BY:

Co-Chair of Advisory Committee

Co-Chair of Advisory Committee

To my family,

for all the love and encouragement they have shown me.

BIOGRAPHY

Jeffrey J. Stuckey was born in East Lansing, Michigan on June 19, 1971. He spent his childhood in Lexington, Kentucky, graduating from Henry Clay High School in 1989. He received his Bachelor of Arts degree in Mathematics from Goshen College in 1993. In 1996, he began the pursuit of a Master of Science in Computer Science at North Carolina State University, graduating in August of 1998.

ACKNOWLEDGEMENTS

This work would not have been possible without guidance from Professor George Rouskas and Professor Yannis Viniotis. I appreciate the generous amount of time that they have spent with me on this work. It has been a pleasure to work closely with both of these talented professors. I would also like to thank Professor Doug Reeves for serving on my committee and for his helpful suggestions.

I would also like to thank my four terrific grandparents, Lou, Wilma, Chauncey, and Pearl, who have shown me much love and taught me important principles such as dedication, integrity, and courage in their daily lives.

Finally, I would like to thank the three most important people in my life: Richard, Judy, and Jon. We are truly blessed to have such a wonderful family and the love and encouragement you have shown me is unparalleled. I thank you with all of my heart.

Contents

List of Tables viii

List of Figures ix

1 Introduction 1
- 1.1 Asynchronous Transfer Mode . . . 1
  - 1.1.1 Quality of Service . . . 2
  - 1.1.2 Traffic Parameters . . . 3
- 1.2 Link Scheduling Algorithms . . . 4
  - 1.2.1 Fixed Priority and Dynamic Priority . . . 4
  - 1.2.2 First In First Out . . . 4
  - 1.2.3 Weighted Round Robin . . . 5
  - 1.2.4 Weighted Fair Queueing . . . 5
  - 1.2.5 Earliest Deadline First . . . 5
  - 1.2.6 Variations and Combinations . . . 6
- 1.3 Introduction to Our Research . . . 6
- 1.4 Thesis Organization . . . 6

2 Overview of Previous Work 7
- 2.1 Earliest Deadline First Research . . . 7
- 2.2 Percentile Analysis . . . 11
- 2.3 Other Relevant Research . . . 11

3 Cell Delay in ATM Networks 13
- 3.1 Call Admission Control . . . 13
- 3.2 The Degree of Multiplexing . . . 14
- 3.3 Leaky Bucket Policing and Shaping . . . 15
- 3.4 ATM Switch Technology . . . 15
- 3.5 Theoretical Maximum Delay Bounds . . . 17
- 3.6 Earliest Deadline First . . . 18
  - 3.6.1 Characteristics . . . 18
  - 3.6.2 Rationale for Choosing EDF for Scheduling . . . 19

4 Simulation Environment **21**
4.1 System Topology . . . 21
4.1.1 Sources . . . 21
4.1.2 Leaky Buckets . . . 26
4.1.3 Multiplexers . . . 27
4.1.4 Links . . . 28
4.1.5 Buffers . . . 28
4.1.6 Switch Fabric . . . 29
4.2 Parameter Set and Values Used in Experimentation . . . 30
4.2.1 Number of Cells . . . 30
4.2.2 Number of Input Ports . . . 30
4.2.3 VC Topology . . . 30
4.2.4 Size of Input Buffers . . . 31
4.2.5 Size of Output Buffer . . . 31
4.2.6 Input and Output Link Speed . . . 31
4.2.7 Source Speeds . . . 32
4.2.8 Packet Size . . . 32
4.2.9 Idle Period Mean . . . 32
4.2.10 Pause Period Mean . . . 32
4.2.11 Number of Packets per Active Period Mean . . . 33
4.2.12 Source Start Times . . . 33
4.2.13 Traffic Mode . . . 33
4.2.14 Cell Delay Variation Tolerance for Peak Cell Rate . . . 33
4.2.15 Cell Delay Variation Tolerance for Sustained Cell Rate . . . 34
4.2.16 Verbose Modes . . . 34
4.2.17 Random Number Seed . . . 34
4.2.18 Derived Values . . . 34
4.3 Parameter Prioritization and Impact . . . 35
4.4 Bin Size Rationale . . . 36
4.5 Worst-Case Cell Delay and The Warm-Up Period . . . 37

5 Simulation Results **38**
5.1 Three State Source Model Experiments . . . 41
5.1.1 Homogeneous Virtual Circuits . . . 41
5.1.2 Heterogeneous Virtual Circuits . . . 47
5.2 Persistent Sources . . . 56
5.2.1 The Aligned Case . . . 56
5.2.2 The Staggered Case . . . 58
5.3 Repetitive Burst Sources . . . 59
5.3.1 The Aligned Case . . . 60
5.3.2 The Staggered Case . . . 62
5.4 MPEG Traces . . . 63

6 Summary and Future Work **66**
6.1 Summary 66
6.2 Future Work 68

Bibliography **69**

A How to Run the Simulator **72**

B Simulator Utilities **78**
B.1 CAC 78
B.2 ExtractColumns 79
B.3 GetPercentile 79

C Simulation Parameters **81**

D MPEG Traces **83**

List of Tables

C.1 Selected simulation input parameters 82
C.2 Output link utilization percentages 82

D.1 MPEG encoder parameters used in generating traces 83

List of Figures

3.1 A generic ATM switch architecture 16

4.1 System topology used in simulations 22
4.2 The three state model for simulating bursty source behavior 23
4.3 Leaky bucket output pattern from a persistent source 26
4.4 Leaky bucket output pattern from a repetitive burst source 27

5.1 Cell delays with 256 VCs on 64 input links 42
5.2 Cell delays with 256 VCs on 2 input links 43
5.3 Tail behavior of cell delays with 256 VCs on 64 input links 45
5.4 Delay percentages with 256 VCs and 97% output link reservation 46
5.5 Delay percentages with 256 VCs and 65% output link reservation 46
5.6 Cell delays with 256 VCs in 2 VC classes on 32 input links 48
5.7 Manipulation of the second burst of cells 50
5.8 Cell delays with 256 VCs in 2 VC classes on 16 input links 51
5.9 Manipulation of the second bounce in cell delays 53
5.10 Cell output times by VC class with 64 input links 54
5.11 Cell output times by VC class with 2 input links 55
5.12 Cell delays with 256 persistent VCs in 2 VC classes on 64 input links in aligned mode 57
5.13 Cell delays with 256 persistent VCs in 4 VC classes on 64 input links in staggered mode 59
5.14 Cell delays with 256 repetitive burst VCs in 2 VC classes on 64 input links in aligned mode 61
5.15 Cell delays with 256 repetitive burst VCs in 4 VC classes on 64 input links in staggered mode 62
5.16 Cell delays with 25 MPEG video sources 65

Chapter 1

Introduction

This chapter provides a brief background on much of the terminology used throughout this document.

1.1 Asynchronous Transfer Mode

Many in the computer networking community have predicted that Asynchronous Transfer Mode (ATM) will be the foundation of choice for most networking applications in the future. Because of its hype and potential, ATM has received a tremendous amount of attention and study in recent years. Whether ATM obtains the sweeping popularity that many predict remains to be seen.

An ATM network is connection-oriented. In contrast to datagram services where packets are sent using a store-and-forward approach, connection-oriented services require that a single logical route be established through the network from source to destination for a particular flow of data. This logical route is referred to as a virtual path (VP). Within each virtual path there exists one or more virtual circuits (VC) which carry the traffic for an individual application from source to destination. In our work we focus on data flows at the VC-level as opposed to the VP-level.

An advantage to a connection-oriented approach is that all packets will travel along the same VC through the network; therefore, packets will arrive at the destination in the order transmitted and the destination will not need to resequence packets. A disadvantage to a connection-oriented approach is that a VC must be established before data may be transmitted which requires some overhead.

In an ATM network, data is packetized into 53 byte units called cells. 48 of these 53 bytes consist of the actual data to be transmitted, called the cell payload. The first 5 bytes of each cell are overhead bytes called the cell header. The cell header is used for VC and VP identification, error checking, and cell loss priority.

These relatively small cells give ATM many advantages over traditional network protocols with larger, variably-sized packets. First, since all of the cells are a small size, we do not have the overhead of fragmenting and defragmenting along the path. Secondly, the modularity of ATM cells allows us to disregard the issue of packet preemption which can result in substantial overhead in a switch. Finally, in the event that a packet must be retransmitted due to cell loss or bit errors, the small cell size results in minimal retransmission overhead.

1.1.1 Quality of Service

One of the strengths of ATM is its suitability for a wide array of traffic types. ATM is engineered for video, voice, and data traffic. Some of these types of traffic have very different requirements. For example, video and voice are often considered real-time traffic; whereas data traffic, such as that being transmitted using the File Transfer Protocol (FTP), is considered non-real-time.

In the transmission of real-time traffic, cells must arrive at the destination by a certain time in order to meet their playback instant—the time at which a cell's data is presented to the user. If a source sending real-time traffic has a cell which incurs a delay such that it misses its playback instant, then this cell is of little or no use to the destination. In contrast, non-real-time traffic does not need to meet playback instants. Even if a source sending non-real-time traffic has a cell whose delay is relatively large, this cell will likely still be of use to the destination.

The type of service requested by a source for its traffic is referred to as its Quality of Service (QoS). Naturally, the better the QoS desired by a user, the more the user will have to pay monetarily for this service. From the network's perspective, a better QoS will result in smaller delays, but use more of the network's resources. In our experiments, we consider various QoS levels represented as different classes of VCs for real-time traffic.

The ATM Forum [3] proposes several different bit rate characteristics that a source may employ when generating traffic. In one such option, a source may generate cells at a

Constant Bit Rate (CBR). With CBR, cell arrival times are deterministic as there exists a constant inter-arrival time between cell transmissions.

The ATM Forum suggests two different types of Variable Bit Rate traffic: one for real-time traffic (VBR-rt[1]) and one for non-real-time traffic (VBR-nrt). In either case, the inter-arrival time varies between cells. Voice and video traffic can be of either the CBR or the VBR-rt variety.

Available Bit Rate (ABR) traffic is a "best effort" service. That is, the network will provide its best effort to service ABR traffic if no higher priority traffic needs service. Unspecified Bit Rate (UBR) traffic is another type of best effort service. Unlike ABR, most traffic parameters are not given to the network for UBR traffic.

Since we are most interested in studying the delays of real-time traffic, our experiments included CBR traffic and VBR-rt traffic. However, the deterministic nature of CBR gave us very uninteresting results and lead us to more extensive experimentation with VBR-rt. Due to their lack of suitability for real-time traffic, we chose not to experiment with VBR-nrt, ABR, or UBR.

1.1.2 Traffic Parameters

ATM traffic is characterized by several parameters on a per-VC basis. The Peak Cell Rate (PCR) is defined as the maximum allowable rate at which a source may transmit. Often, the PCR is equal to the speed of the transmission medium. In specifying a PCR with the network, a source promises that it will not send cells at a rate any higher than the PCR during any point of transmission.

The Sustained Cell Rate (SCR) is defined as the average rate at which a VC is allowed to send cells through the ATM network. In specifying an SCR with the network, a source promises that it will not send cells at a rate any higher than the SCR during a given interval of time. Of course, a source may send cells at a lesser rate than the SCR during this time interval. For CBR traffic the SCR is equal to the PCR and for VBR traffic the SCR is strictly less than the PCR.

A group of cells that is transmitted consecutively and relatively rapidly after each other is commonly referred to as a "burst" of cells. A burst is not easily quantified and is a notion, not a distinct measure. However, there exist several methods of estimating

[1] In future chapters, we will reference VBR-rt with simply the term VBR.

burstiness. The Maximum Burst Size (MBS) is defined as the greatest number of cells that can be transmitted at the negotiated PCR. The Burst Tolerance (BT) is defined as the length of time at which a source may be allowed to transmit at PCR.

The Cell Delay Variation Tolerance (CDVT) is a lower bound on the amount of time permissible between consecutive cell arrivals still considered conforming to either the PCR or SCR. In regards to the PCR, the CDVT is denoted CDVT_{PCR}. Similarly, in regards to the SCR, the CDVT is denoted CDVT_{SCR}.

Of the above parameters, only PCR and CDVT_{PCR} are applicable to CBR traffic; however, each of the above parameters is required for VBR-rt traffic.

1.2 Link Scheduling Algorithms

The question of how to best schedule cell departures from ATM switches is not clear-cut and may depend upon the traffic characteristics. Many scheduling algorithms have been proposed in the literature. In this section we discuss some of the more well-known scheduling algorithms.

1.2.1 Fixed Priority and Dynamic Priority

Scheduling algorithms may be either fixed priority or dynamic priority. Intuitively, in fixed priority scheduling, once a priority has been assigned to a cell, this priority will not change regardless of subsequent cell arrivals. On the contrary, dynamic priority scheduling allows a cell's priority to change in time depending on subsequent arrivals. A fixed priority scheme does not involve the overhead of changing priority assignments and is relatively easy to implement; however, it does not minimize delays as well as dynamic priority schemes in most cases.

1.2.2 First In First Out

First In First Out (FIFO) scheduling is also known as First Come First Served (FCFS) scheduling. In this fixed priority scheduling method, cells depart the switch in the same order as they arrived at the switch. In this method, priority is a function of cell arrival instants. Clearly, this is undesirable when different levels of QoS are requested. An advantage to this scheme is that it is very easy to implement in the switch hardware.

1.2.3 Weighted Round Robin

The Weighted Round Robin (WRR) algorithm is a fixed priority approximation of the purely theoretical General Processor Sharing (GPS) algorithm. In GPS, infinitesimally small portions of packets are served on a per-VC basis such that each VC can be served at least once in any given interval of time. In GPS, if a VC has no cell to be transmitted, then the amount of time that would have been dedicated to serving this VC is equally shared among the VCs that do require service.

In WRR, weights may be assigned to connections, thus allowing different priority levels. In WRR, each VC is served sequentially, where the length of its service time is proportional to its assigned weight. WRR only provides fairness during an interval of time greater than the time required to service each VC. If there are many VCs with long service times, other VCs will have long wait times. WRR is appropriate for fixed size packets, but not ideal for variable sized packets since WRR must know a VC's average packet size at connection establishment.

1.2.4 Weighted Fair Queueing

Like WRR, the Weighted Fair Queueing (WFQ) algorithm is another fixed priority approximation of GPS. Also like WRR, WFQ allows assigning weights to VCs to provide different levels of service. In WFQ, a packet's estimated finish service time is calculated according to GPS. Then this service time and the VC's weight is used in prioritizing packets for service. WFQ provides reasonable fairness to VCs and does not require knowledge of average packet sizes; however, the large number of calculations required cause WFQ to have a high degree of computational complexity.

1.2.5 Earliest Deadline First

The Earliest Deadline First (EDF) algorithm is the focus of our research. With this dynamic priority algorithm, each cell is assigned a deadline upon arrival at the switch. At any instant in time, the cell with the most immediate deadline is served. EDF is relatively easy to implement, but suffers from complexities at call set-up time. A more in depth discussion of the EDF algorithm is given in Section 3.6.

1.2.6 Variations and Combinations

Many variations of the above algorithms have been proposed in the literature. Further, combinations of these algorithms are also possible. For example, some commercial ATM switches have implemented head of line priorities in combination with WRR scheduling.

1.3 Introduction to Our Research

In this work we examine worst-case cell delays in an ATM switch. Because of EDF's delay optimality and its relatively low degree of implementation complexity, we chose EDF for study in favor of other scheduling algorithms.

Our focus is in researching the tail behavior of cell delay functions under a variety of worst-case conditions. To accomplish this, we have developed a simulation program with which we analyze cell delays using a variety of different source and leaky bucket behaviors.

We observe that worst-case cell delays increase as the degree of multiplexing decreases. We also observe that worst-case cell delays are heavily dependent on the source and leaky bucket models used. Finally, we show that worst-case cell delays increase when sources are aligned to transmit simultaneously.

1.4 Thesis Organization

This thesis is organized as follows. In Chapter 2, we provide a brief overview of other research related to our own. In Chapter 3, we discuss the principles which contribute to cell delays in ATM networks. In Chapter 4, we discuss in detail our simulation environment and the simulator program. In Chapter 5, we discuss the most interesting findings from our experimentation. Finally, in Chapter 6, we summarize our work and discuss some possible future research topics.

Chapter 2

Overview of Previous Work

Much research has been devoted to the Earliest Deadline First scheduling algorithm. However, to our knowledge, we are the first to perform an analysis of ATM switch cell delays using the EDF scheduling algorithm. In this chapter we give an overview of the research most relevant to our own.

2.1 Earliest Deadline First Research

The ground-breaking research on EDF was performed by Liu and Layland [15]. In their work they describe a real-time dynamic scheduling algorithm that they call the Deadline Driven Scheduling algorithm. Currently in the literature, this algorithm is referred to as the Earliest Deadline First (EDF) algorithm. We also use this convention in further references.

While Liu and Layland's research gives us the foundations of the EDF algorithm, their research differs from ours in several ways. First, their work is based upon single processor task scheduling, whereas ours deals with scheduling cell transmissions from the input ports to an output port of an ATM switch. In their studies, tasks are periodic, which means that they must be repeated at regular intervals. In most of our simulations, cell arrivals are aperiodic with varied inter-arrival times. With a purely periodic task set a schedule can be established in advance; however, in our aperiodic cases we must make scheduling decisions at run-time.

Their work also differs from ours in that they assume that a task may be preempted, i.e. interrupted, by a higher priority task. Liu and Layland assume that there

is no overhead in context switching between tasks during preemption and that preempted tasks resume execution without loss of work. We do not allow cell preemption in our simulations. Because of uniform cell sizes, preemption simply does not make sense in an ATM environment.

In their paper, Liu and Layland show a couple of important properties of this algorithm which we also observe. First, they show that when a set of tasks is scheduled according to EDF, there is no idle time prior to an overflow [15]. In other words, the EDF scheduling algorithm is work-conserving in the sense that as long as there are tasks to be scheduled (or cells to be transmitted in our case), the EDF algorithm will schedule the tasks (cells) back-to-back. If a deadline is missed, then the processor (ATM switch in our case) was not idle just prior to this. In our research, deadlines are not missed since the Call Admission Control algorithm will not over allocate the output link and since we have ample buffering inside the switch.

Secondly, Liu and Layland show that EDF is an "optimal" scheduling algorithm. That is, if a set of tasks is schedulable (i.e. there exists a schedule such that all tasks meet their respective deadlines), then the EDF scheduling algorithm will be able to schedule the tasks. This is very important to our study and one of the reasons we choose to study EDF instead of other scheduling algorithms. Largely due to the optimality property and the work-conserving nature of EDF, we expect that EDF will provide favorable delay guarantees.

Another important work in the EDF domain was conducted by Georgiadis *et al.* [7]. In their research they investigate delay and buffer requirements on a single link. They use non-preemptive EDF (NPEDF) while considering three methods of buffer allocation: flexible (FL), fixed (FI), and semi-flexible (SE). In flexible allocation, any buffer space may be allotted to any arrival stream. In fixed allocation, each class of traffic has a reserved amount of buffer space which is assumed to be sufficient for all input traffic rates and bursts. In semi-flexible buffer allocation, there is a fixed amount of buffer space allotted for each class of traffic which may not be changed after the first packet arrives. This space may depend on traffic characteristics.

In their analysis, B represents the minimum buffer size for each allocation method, N is the number of traffic streams, L_{max} is the maximum packet length in bits, and σ_i is the burst size of the i^{th} traffic stream.

They propose the following minimum buffer sizes for each buffer allocation method.

For flexible allocation (FL),

$$B_{\mathrm{FL}}^{lower} = NL_{max} + \sum_{i=1}^{N} \sigma_i.$$

For fixed allocation (FI),

$$B_{\mathrm{FI}}^{lower} = 2NL_{max} + \sum_{i=1}^{N} \sigma_i. \tag{2.1}$$

And for semi-flexible allocation (SE),

$$B_{\mathrm{SE}}^{lower} \geq (2N-1)L_{max} + \sum_{i=1}^{N} \sigma_i.$$

Georgiadis *et al.* [7] also provide the following upper bound on buffer size required for the fixed allocation method:

$$B_{\mathrm{FI}}^{upper} \leq (2N+1)L_{max} + 2\sum_{i=1}^{N} \sigma_i. \tag{2.2}$$

Our buffer implementation falls into both their flexible allocation and fixed allocation categories. In our switch we have a single output buffer into which any VC's cells may be inserted into any position. This falls into the flexible allocation category. Also, in our switch we have a separate input buffer for each VC which is only stores cells for that VC. Thus our input buffers are examples of fixed allocation.

We note that because they assume that the entire burst arrives instantaneously at the buffer, the summations in each of the equations presented here cause these to be extremely conservative bounds.

In the flexible allocation method, the lower bound on buffer size holds for any work-conserving policy, such as EDF. However, it is important to note that Equation 2.1 only holds for Rate Proportional Processor Sharing (RPPS) and not for EDF. Although the minimum buffer size for EDF is lower than that given in Equation 2.1, this bound can be used for EDF, because of its overly conservative nature. Equation 2.2 holds for both RPPS and EDF.

A discussion of how we relate Equations 2.1 and 2.2 to our choice of input buffer sizes is given in Section 4.2.4.

Georgiadis *et al.* [7] continue by giving some schedulability conditions showing that NPEDF is delay-optimal among all non-preemptive scheduling policies. They go on to show that with NPEDF for semi-flexible buffer allocation with N traffic streams into the

system, a switch needs buffer space on the order of N^2 per traffic stream. Next, they design policies which reduce buffer requirements and delay at the expense of reducing the number of traffic streams able to be served.

In a paper on end-to-end delay bounds, Goyal *et al.* [10] consider a single data flow, f, which conforms to a leaky bucket with parameters (σ_f, r_f), where σ_f is the MBS of the flow and r_f is the rate of the flow.

In their delay formula, d_f^j is the end-to-end delay of the j^{th} packet on flow f, K is the number of servers on the path of the flow, l_f^n represents the length of the n^{th} packet, β^n represents the time to transmit the n^{th} packet, and $\tau^{n,n+1}$ represents the propagation delay between the n^{th} server and the destination. The end-to-end delay of a packet is given by,

$$d_f^j \leq \frac{\sigma_f + (K-1)max_{n\in[1..j]}l_f^n}{r_f} + \sum_{n=1}^{n=K}(\beta^n + \tau^{n,n+1}). \tag{2.3}$$

This theoretical maximum delay bound is further discussed and modified to fit our simulation environment in Section 3.5.

In order to understand the work performed by Georgiadis *et al.* [8, 9] we must first introduce the concept of traffic reshaping. Traffic reshaping is a means of controlling jitter, the variation of delays, across an ATM network which involves buffering cells at the output of each intermediate switch. Clearly, a VC's MBS and SCR may be different when observed at a switch's output and it's input, depending on other traffic also being served by the switch. If the SCR at the output of a switch is greater than at the input, this could result in the next downstream switch receiving a greater SCR for this VC than was expected. Since most EDF feasibility schedules rely heavily on a VC's SCR, the feasibility schedule of the next downstream switch could be violated if no traffic reshaping is performed. Another advantage of traffic reshaping is that by delaying cells at the output of a switch, buffer requirements for subsequent switches are reduced. However, implementation of traffic reshapers in switches is non-trivial.

One initially counter-intuitive notion presented in [8, 9] suggests that the increase in cell delays caused by buffering at the output of each switch for traffic reshaping purposes does not cause an increase in end-to-end cell delays. This result can be understood when we consider that although cell delays at the output of a switch will likely increase, traffic reshaping offsets these cell delays by decreasing cell delays inside the downstream switches

along a VC's path. Thus the end-to-end delay is unaffected.

In our study, we only consider the single node case; however, the work in [8, 9] leads us to believe that we could use the EDF scheduling algorithm in the multiple-node case by placing traffic shapers at the output of each switch along a VC's path.

2.2 Percentile Analysis

Several researchers have performed various percentile analyses; however, each of their works fundamentally differ from our research. Liang and Yuang [13] consider EDF in ATM switches; however, their work focuses on percentile analyses of cell inter-departure times, whereas our research considers cell delays. Schulzrinne *et al.* [19] consider network percentile delays; however, their focus is on a hop laxity scheduling algorithm, whereas we consider EDF. Both Hirano [11] and Zein [22] consider ATM cell delays; however, both use FIFO ordering at all buffers inside of the switch. In another paper, Chao [2] gives a very brief percentile analysis of 4 VCs; however, he uses a different "on-off" source model than ours.

2.3 Other Relevant Research

Much other research has been conducted on related topics, such as WFQ, call admission control, and source modeling.

A popular scheduling technique implemented in ATM switches is WFQ (see Section 1.2.4). Delay bounds for WFQ can be found in the work of Parekh and Gallager [16, 17] for both the single and multiple node cases. Many variations of WFQ exist including WF^{2}Q, as outlined by Bennett [1].

Lieberherr *et al.* [14] derive a schedulability condition for NPEDF implemented in a single node. In their analysis, they consider flows f and k which are elements of the set of $\mathcal{N}$ connections which may be affected by the constraint function, A^*. A_f^* serves as a traffic policer and rate controller for flow f. Let s_f^{max} denote the maximum packet transmission time on flow f. Let the number of arrivals on flow f with requested delay d_f^{req} which occur before or at time t be given by $\sum_{f \in \mathcal{N}} A_f^*(t - d_f^{req})$. Then if $\forall t \geq d_1^{req}$, where d_1^{req} is the smallest delay requested, the EDF schedulability test is given by:

$$t \geq \sum_{f \in \mathcal{N}} A_f^*(t - d_f^{req}) + \max_{k, d_k^{req} > t} s_k^{max}, \tag{2.4}$$

where $\max_{k, d_k^{req} > t} s_k^{max} \equiv 0$ for $t > \max_{k \in \mathcal{N}} d_k^{req}$. Note that the term $\max_{k, d_k^{req} > t} s_k^{max}$ simply means the maximum time to transmit a packet for flow k, where the requested delay of k is greater than t. This term provides for the non-preemptive nature of NPEDF and is equal to either 0 or the time to transmit a cell in an ATM environment. Finally, the term $A_f^*(t - d_f^{req})$ tests the schedulability of the SCRs for flow f.

Elsayed and Perros [5] give schedulability conditions for the multiple node case. In their analysis they assume we have VCs i and j with requested delays d_i^{req} and d_j^{req}, respectively, where $d_i^{req} < d_j^{req}$ if $i < j$. Let σ_i represent the MBS of flow i and r_i represent its average rate. Then $Pmax_k$ is the maximum packet size from connection k. Finally, given that $\sum_{i=1}^{N} r_i < C$, where C is the link capacity, the schedulability condition is given by:

$$d_j^{req} \geq \frac{\sigma_j + \sum_{i=1}^{j-1} (\sigma_i - r_i d_i^{req}) + max_{k>j} Pmax_k}{C - \sum_{i=1}^{j-1} r_i}. \tag{2.5}$$

Firoiu [6] discusses an improved CAC for EDF. Kawasaki [12] proposes a new CAC algorithm, but for a variant of Round Robin scheduling.

The source model that we use for our research on EDF is part of the ATM Forum [3] standard and is also used by Wright [21]. Wright gives tail delay analyses; however, he considers WFQ scheduling.

Variants of EDF have also been proposed in the literature. Cruz [4] discusses an EDF variant, SCED+. Spuri [20] also discusses EDF variants. Zhang [23] performs percentile analyses; however, uses a different scheduling algorithm and source model than ours.

Chapter 3

Cell Delay in ATM Networks

While there are many factors which contribute to a requested QoS such as cell loss rate, bit error rate, etc., we choose to focus our study on the most critical factor for real-time applications—cell delay. In this chapter, we explore some of the factors most directly related to cell delay in ATM networks.

3.1 Call Admission Control

When a source makes a request to establish a VC through the network, a decision must be made by the network regarding whether or not to accept the request. Upon considering this, the network must concern itself with two issues. First, the network must have sufficient resources available to grant the requested QoS desired by the source. Secondly, if this request is granted, the QoS of existing VCs which use any of the switches through which the new VC will be established must not be compromised. If both of these conditions can be met, then the connection is accepted; otherwise, it is rejected. This decision-making process is referred to as call admission control (CAC) or, alternatively, connection admission control.

The network uses a CAC algorithm to determine how much bandwidth to reserve for a given VC on each link used by that VC. The network may choose to not overallocate the link bandwidth. In this case, deterministic delay guarantees can be given. That is, the network can give a source a 100% guarantee that all of its traffic will be delivered within its delay bound.

Alternatively, the network may choose to overallocate its link bandwidth. In this

case, the network provides a source percentile guarantees regarding the delays of that source's traffic. For example, the network may be able to guarantee that at least 99% of a source's traffic will meet the given delay requirements. For real-time traffic which can sustain minor delay violations, a percentile guarantee may be acceptable. Percentile guarantees are a more inexpensive option for a user than deterministic guarantees, since the network will be able to accept more connections; hence, reduce the cost on a per-VC basis.

Both schedulability condition algorithms, Equations 2.4 and 2.5, discussed in Section 2.3 take a user's maximum delay request, d^{req}, into consideration when deciding whether or not to accept a connection. In our CAC algorithm we make a simplifying assumption and do not reject connections on this basis alone. In other words, given that d^{max} is the theoretical maximum delay bound for EDF scheduling, we assume that each VC makes a delay request d^{req} such that $d^{req} = d^{max}$. Thus we will accept N connections as long as their cumulative SCRs do not exceed the input and output link capacity, C. Thus the following must be satisfied at all times:

$$\sum_{i=1}^{N} \mathrm{SCR}_i \leq C. \tag{3.1}$$

We note that with the non-preemptive Packetized General Processor Sharing (PGPS) scheduling algorithm, Equation 3.1 is used in determining whether or not to accept a VC. We use this same equation in our work, since the theoretical maximum delay bound for EDF (which we are using as our d^{req} value) is not greater than that for PGPS. In other words, if PGPS accepts a VC then our CAC for EDF will also accept that VC.

3.2 The Degree of Multiplexing

Another important factor in studying cell delay results from the "degree of multiplexing". With the term degree of multiplexing we mean the greatest number of VCs multiplexed onto a single input link into our switch.

With a relatively low degree of multiplexing, sources generally will not incur much delay at a multiplexer in competition with other sources multiplexed onto that same input link. However, in this scheme sources may incur relatively large delays inside of the switch while awaiting transmission onto the output link.

With a high degree of multiplexing, sources may incur large delays at a multiplexer

in competition with other multiplexed sources; however, inside the switch they will have smaller delays. Of course, the above two arguments assume that all other parameters are kept constant.

It is important to note that from a user's perspective, it is irrelevant whether the bulk of delay occurs at a multiplexer or inside a switch. The user's only concern is the end-to-end delay. However, this is not the case for the network. Since the multiplexers physically reside outside of the switch, the network is not concerned with multiplexing delays, rather only the delays while a cell is inside of the switch. The key notion here is that if the delay of a traffic flow can be diverted from a switch to a multiplexer, then the network can accept more VCs which reduces each user's cost while still delivering the same end-to-end QoS.

3.3 Leaky Bucket Policing and Shaping

A leaky bucket is a mechanism which may alter cell traffic patterns entering a network node. A leaky bucket typically resides at source, or at the output of an intermediate network node, or both. A leaky bucket serves two purposes, namely traffic policing and traffic shaping.

In its traffic policing role, a leaky bucket insures that a source does not exceed its negotiated traffic contract with the network, whether intentionally or unintentionally. From the network's point of view, a source cannot be trusted. Without traffic policing, a single misbehaving source could severely degrade the QoS of all other behaving VCs using the link.

In its traffic shaping role, a leaky bucket "smoothes" traffic that is inherently bursty. The term smoothing refers to spreading out a burst of cells over time. Clearly, network delays can be minimized when traffic is more regular.

Many leaky bucket variants and related traffic shapers and traffic policers exist. We chose to use the leaky bucket model recommended by the ATM Forum in [3] which suggests a virtual scheduling algorithm based on the Generic Cell Rate Algorithm (GCRA).

3.4 ATM Switch Technology

In this section we discuss the characteristics of ATM switches in general, with emphasis on the properties that we chose to implement. A discussion regarding our imple-

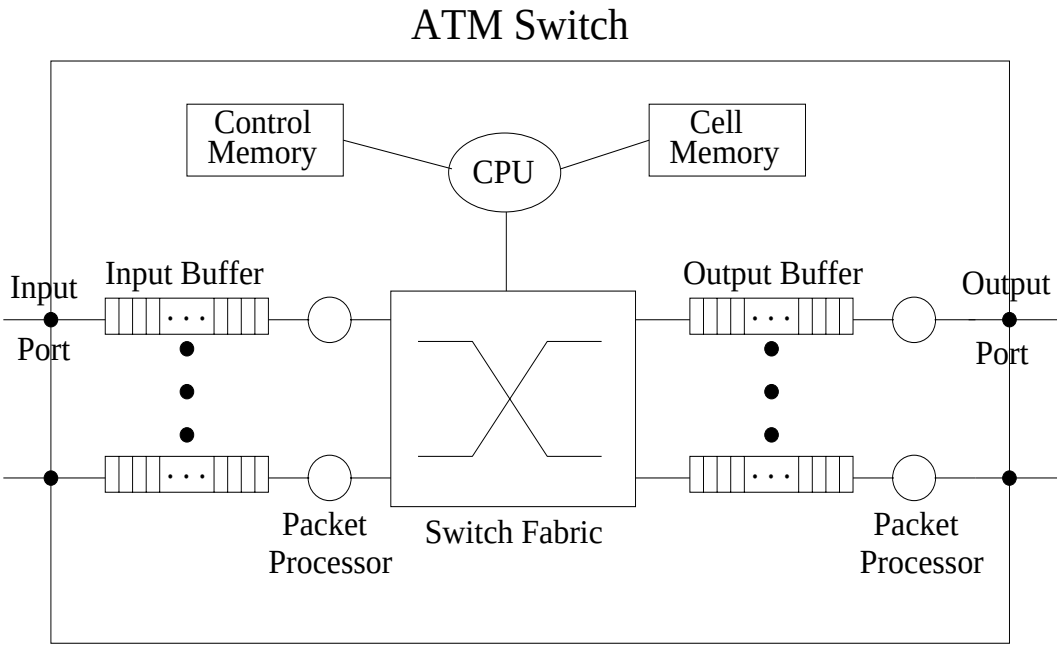

Figure 3.1: A generic ATM switch architecture

mentation specifics can be found in Chapter 4.

ATM networks typically involve many ATM switches along the path from source to destination. For simplicity, we focus on observing the delays in a single-node network topology. The multiple ATM switch case is reserved for future work.

A typical ATM switch architecture is shown in Figure 3.1. In a single-node network all traffic passes through the common ATM switch which connects each of the input links[1] to each of the output links. Typically, the number of input links and the number of output links are multiples of 2, most commonly 32 or 64, and are in most cases equal to each other. Of course, multiple VCs may be multiplexed onto each input and output link and input links are often shaped and policed by leaky buckets.

The switch fabric (see Figure 3.1) is the architecture by which cells are transferred from an input buffer to an output buffer. This switch fabric may be blocking or non-blocking. In a blocking switch fabric, two cells simultaneously bound for the same output port will result in one of the cells being dropped. A non-blocking fabric is designed such

[1] We use the term "link" to designate the transmission medium on which a cell travels and the term "port" to denote a link's interface to the switch.

that only one cell may be bound for a given output port at any point in time. For simplicity, in our study we assume a non-blocking fabric. The amount of time it takes for a cell to propagate through the switch fabric can be considered constant.

Because all cells have a relatively small and unit length, an ATM switch will not preempt a cell once it has begun transmission. In general, preemption involves significant overhead in context switching to the next packet and saving the state of the preempted packet if non-duplication of work is required. This overhead is clearly not justified for the small cell transmission times that ATM produces. Even if preemption was implemented in ATM switches, the decrease in delay for the preempting cell would be at most one cell transmission time—a very minimal decrease.

Inside of a switch, there typically exists one input buffer per input link and one output buffer per output link (see Figure 3.1). These buffers allow cells to be queued for transmission while waiting for other cells to depart. Also, each link contains a packet processor whose function is to serve the cells from their respective buffer.

The operation of the switch is managed by the Central Processing Unit (CPU). It is the responsibility of the CPU to obey the scheduling algorithm in keeping cells moving through the switch fabric. The CPU uses the control memory and cell memory in accomplishing this (see Figure 3.1). The control memory stores a linked list of tasks which the CPU executes. The cell memory provides temporary storage for cells.

3.5 Theoretical Maximum Delay Bounds

A theoretical maximum delay bound, for real-time traffic, is the greatest amount of delay permissible for a VC's cells to still be of use to the destination, i.e. such that a cell does not miss its playback instant. Theoretical maximum delay bounds may differ on a per-VC basis.

As we discussed in Section 2.3, Goyal [10] provides a means for calculating the theoretical maximum delay bound for a flow. Equation 2.3 is repeated here for reference,

$$d_f^j \leq \frac{\sigma_f + (K-1)max_{n\in[1..j]}l_f^n}{r_f} + \sum_{n=1}^{n=K}(\beta^n + \tau^{n,n+1}).$$

Since we are only considering the single node case, $K = 1$ for our experiments. Since we ignore propagation delays, we ignore $\tau^{n,n+1}$. Further, for EDF β^n is defined to be

the amount of time required to transmit a cell at link speed, t_{cell}. Therefore, the term $\sum_{n=1}^{n=K}(\beta^n + \tau^{n,n+1}) = t_{cell}$.

We are left with a much simplified version of the above formula, namely,

$$d_f^j \leq \frac{\sigma_f}{r_f} + t_{cell}. \tag{3.2}$$

We note that because EDF is a dynamic priority scheduling algorithm, its theoretical maximum delay bound is impossible to calculate. Equation 2.3 holds for PGPS scheduling. We use this PGPS theoretical maximum delay bound as an approximate EDF theoretical maximum delay bound, since we expect these bounds to be very close. This theoretical maximum delay bound was used in all of our experiments and was never exceeded.

3.6 Earliest Deadline First

A brief background on EDF can be found in Section 1.2.5. In this section we provide a more in depth discussion on the characteristics of EDF and our reasons for studying this algorithm.

3.6.1 Characteristics

In Section 2.1 we reviewed Liu and Layland's [15] early work on EDF. In their paper, they showed that EDF is an optimal scheduling algorithm in that if a task set is schedulable, then EDF is guaranteed to be able to schedule it. They also showed that if a deadline is missed, then there was no idle period just prior to this. In this sense EDF is work-conserving.

Further, EDF is a dynamic priority scheduling algorithm. As such, the priority of a cell may change in time. Associated with each VC using the switch is a theoretical maximum delay bound, d_j^{max}. d_j^{max} is given such that each cell of a particular VC, VC_j, will not incur a greater delay than d_j^{max}. If a cell on VC_j has a delay greater than d_j^{max}, then, assuming we want to provide 100% guarantees, our CAC has failed us by overallocating the output link.

In measuring the delay of cells while in a switch, each cell has associated with it an arrival time, a_j, and a deadline, D_j. The theoretical maximum delay bound for a cell's

VC is added to its arrival time to compute the cell's deadline. Thus for VC_j,

$$D_j = a_j + d_j^{max}. \tag{3.3}$$

It is important to note that the deadline assigned to a cell does not change in time. However, its priority relative to other cells may change in time depending on the priorities of newly arriving cells. Note that cells with smaller deadline values, are those cells which are closer to their theoretical maximum delay bound, and thus have higher priority in the switch.

The EDF scheduler operates such that at any given instant of time, the cell with the smallest deadline is served, i.e. transmitted on a switch's output link. The exception being when a cell is in the process of being transmitted onto the output link and a higher priority cell arrives at the switch.

In a multi-node network, traffic reshaping is often performed at the output of each node in order to make the input of the subsequent node more regular. Further discussion of traffic reshaping is given on p. 10.

In commercial ATM switches the WFQ scheduling algorithm (see Section 1.2.4) and the FIFO scheduling algorithm (see Section 1.2.2) have achieved more popularity than has EDF. One of the reasons for this is the complexity of the CAC algorithms for EDF in comparison to those for WFQ and FIFO. Despite these obstacles with CAC complexity, EDF scheduling is much better suited for different QoS levels than is FIFO. Further, EDF is a much simpler algorithm to implement in the hardware inside of an ATM switch than is WFQ.

3.6.2 Rationale for Choosing EDF for Scheduling

We chose to research the behavior of the EDF scheduling algorithm in an ATM switch for several reasons. First and foremost, as was mentioned in Section 2.1, EDF is an optimal scheduling algorithm for a single node. As such, EDF can provide a feasible schedule if such a schedule exists. Clearly, no other scheduling algorithm can improve on EDF's ability to schedule tasks.

Also, EDF has some clear advantages over WFQ—its main competition in ATM. Although Elsayed and Perros [5] show that a CAC algorithm for EDF may admit fewer connections than a CAC algorithm for WFQ, the hardware complexity for WFQ is such

that it may limit the number of connections which WFQ, in theory, could accept. A large advantage of the EDF algorithm is that it is much simpler to implement in a switch's hardware than WFQ.

Chapter 4

Simulation Environment

The event-driven simulation program was written in C++ and compiled on UNIX Solaris SPARCstation 4 terminals. The simulator consists of 15 C++ classes and contains approximately 5,000 lines of code. Instructions on running the simulator and its utilities are provided in Appendices A and B, respectively. All of the simulations were run on SPARCstation 4 machines in the public laboratories of North Carolina State University.

4.1 System Topology

In this section, we describe the details of the single-node network we are modeling.

4.1.1 Sources

In our model each of a number of sources in the system transmits real-time traffic to a single destination (see Figure 4.1). Of course, in an actual implementation, an ATM switch will connect many permutations of source-destination pairs. However, by focusing on only the traffic bound for a single destination, we bring our attention to the increase in cell delays due to competing sources.

We also assume that before the simulator has begun all of the sources have successfully established exactly one virtual circuit to the common destination. Thus, all sources are initially prepared to send their data cells (as opposed to connection establishment overhead cells). Further, we make the assumption that the connection set-up time is constant for all VCs, and since we are only interested in data cell delay, the connection set-up time can be eliminated from further consideration.

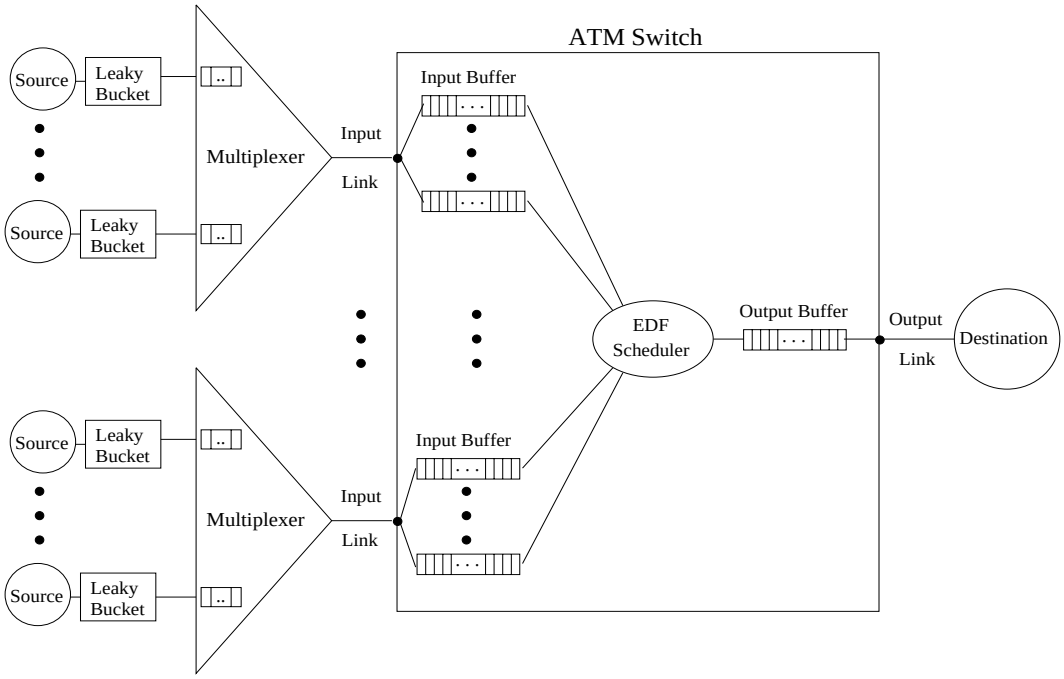

Figure 4.1: System topology used in simulations

Similarly, we ignore propagation delays from each source to the ATM switch and also from the ATM switch to the destination. We assume that these propagation delays are the same for all sources; hence, they can be disregarded as a delay constant.

We assume that there is a finite amount of time between cell transmissions from a source. In this light, we assume that sources never close their virtual circuit with the destination. In other words, a source will not finish sending cells for the duration of each simulation. Further, no new VCs may be established while the simulator is in progress. These simplifying assumptions give us more uniform behavior of the sources.

Each source may transmit according to either the CBR or VBR modes as described in Section 1.1.2. A discussion of each mode follows.

CBR Traffic

A source transmitting at constant bit rate is deterministic. The PCR for CBR sources is determined by an input parameter.

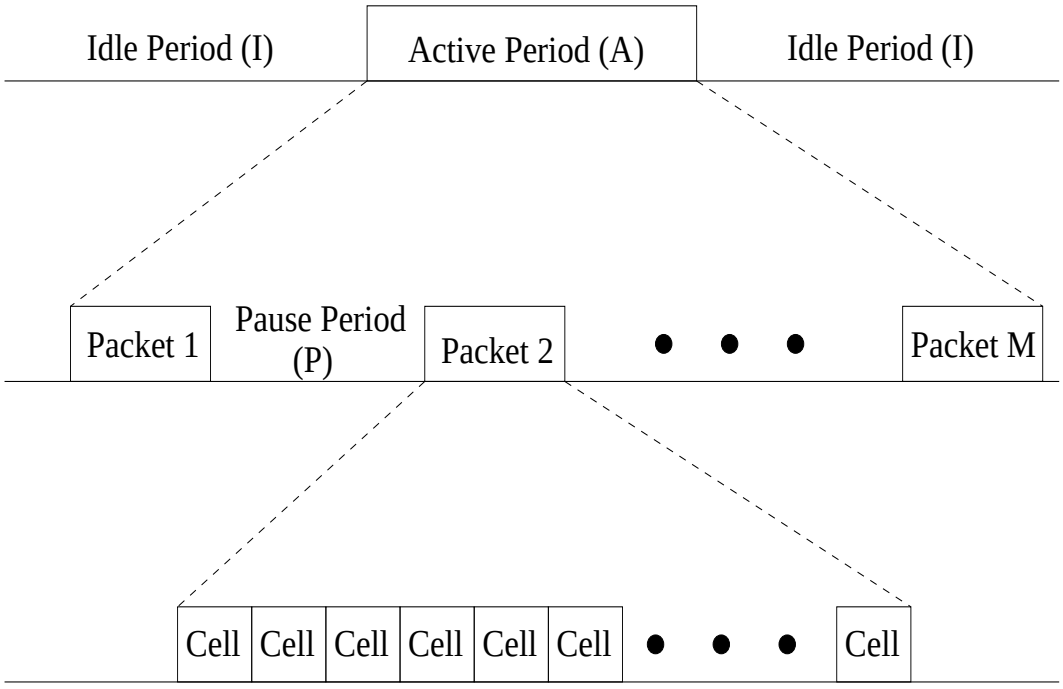

Figure 4.2: The three state model for simulating bursty source behavior

VBR Traffic and The Three State Source Model

The ATM Forum [3] recommends the following three state source model for simulating VBR traffic. The model introduces the following concepts: active periods (A), idle periods (I), pause periods (P), the number of packets per active period (M), and the size of packets (S). For reference see Figure 4.2.

Each source alternates between active and idle states. While in the active state, a source is being given data from an application to transmit. When in the idle state, the source is waiting for the next units of data from an application. The length of an idle period is randomly distributed according to the exponential distribution with mean, $\overline{I}$, given as an input parameter. Each idle period is calculated according to the formula:

$$I = \overline{I} * -\ln(random[0..1]). \tag{4.1}$$

The packet size, S, is given as an input parameter. Given that t_{packet} is the amount of time required to transmit a packet, the length of each active period is determined by the

following expression:

$$A = MSt_{packet} + (M-1)P.$$

M, the number of packets in the active period, is determined by a geometrically distributed random variable. The mean of this distribution, $\overline{M}$, is provided to the simulator as an input parameter on a per-VC class basis. The value of M is computed according to the formula:

$$M = \left\lceil \frac{\ln(random[0..1])}{\ln\left(1 - \frac{1}{\overline{M}}\right)} \right\rceil .$$

While in the active state, a source packetizes data for transmission. Each packet is then divided into 48 byte units for placement into the payload of an ATM cell. Padding is assumed for the last cell in a packet if the packet size is not evenly divisible by 48 bytes. These cells are then sent from the source without any inter-cell delay. Since it is very small and constant for all sources, we neglect the time it takes a source to divide a packet into ATM cells and add the ATM cell header.

In between generating each of the M packets, a source will pause for time P, where P is a randomly distributed variable according to the exponential distribution. The pause period takes into consideration the time that sources spend accumulating data into a packet. Pause periods are generated using the following equation where $\overline{P}$ is the distribution mean:

$$P = \overline{P} * -\ln(random[0..1]).$$

Intuitively, pause periods should be less than idle periods on average. Our choosing of their respective mean values reflects this intuition.

In initializing the three state source model, we experimented with starting a source at the beginning of both an active period and an idle period. When we started all sources on active periods, each source sent the first cell from the first packet in the active period to its leaky bucket at a global time of zero. We refer to this mode of initialization as "aligned" mode, since all sources are synchronized to begin initial transmissions at the same instant in time.

In other experiments, we started sources at the beginning of an idle period. In this case, at simulation initialization each source waits for a random idle period (see Equation 4.1) before becoming active and transmitting its first burst of cells. We refer to this mode of initialization as "staggered" mode, since all sources begin initial transmissions at

different instants in time. We have observed that "staggered" mode experiments produce much more favorable worst-case cell delays than do similar "aligned" mode experiments.

An advantage of the three state source model is that depending on the distributed variable means, the sources produced are very bursty. In measuring the ratio of observed peak cell rate to mean cell rate over 1 second intervals, we found that for typical input parameters, this ratio has maximum values on the order of 100. This means that these simulated sources can be extremely bursty. As a comparison, peak to mean ratios for the MPEG video traces we used were typically on the order of 10 to 20 (see Appendix D).

A disadvantage of the three state model can be found in the choice of the exponential distribution for generating random variables for the idle and pause periods. The exponential distribution employs the "memoryless" property wherein lies the principle that each random variable is independent of its predecessor. This simplifying assumption; however, is not the case found in actual VBR sources. Typically, idle periods will be correlated, as a source will likely have smaller idle periods when the transmitting application has data to send and likely have longer idle periods when the application needs to wait for user input, disk I/O, etc. These events are likely to be correlated.

Despite this disadvantage, we chose this three state model because of its ease in implementation as well as its conformance to the ATM Forum's recommendation for source modeling [3].

The Persistent Source Model

In addition to our experiments with the three state model, we have also experimented with two source models with which we obtained worst-case leaky bucket output. We call the first such model the "persistent" source model. In this model the leaky buckets accept individual cells as soon as their parameters allow. In the persistent source model, sources transmit cells continuously at a rate of PCR. Thus, the vast majority of cells are dropped by the leaky buckets.

The pattern of cells output from the leaky buckets is shown in Figure 4.3. In this figure, we note that the first MBS cells are leaky bucket conformant. After the last cell in the first burst, cells are leaky bucket conformant at a rate of SCR.

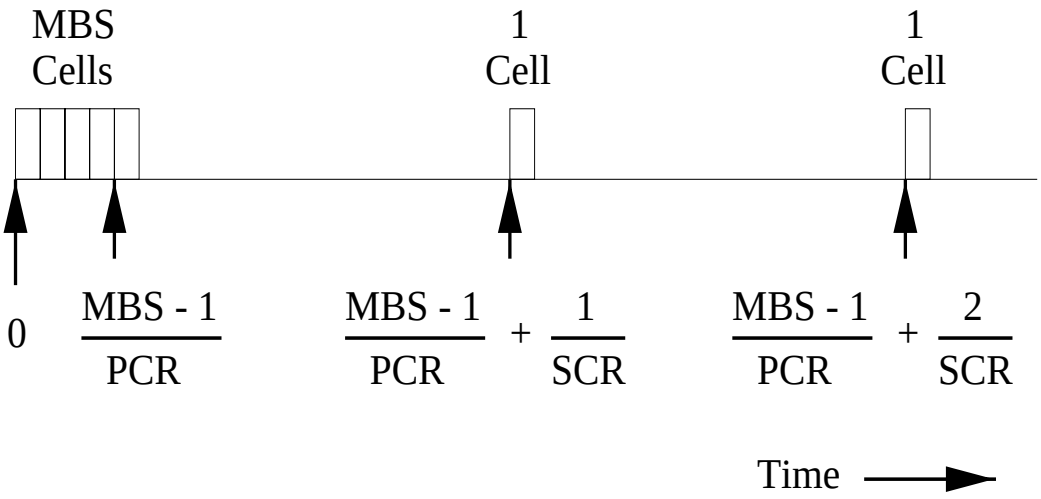

Figure 4.3: Leaky bucket output pattern from a persistent source

The Repetitive Burst Source Model

With the "repetitive burst" source model we experimented with another worst-case model in terms of leaky bucket output. In this model, the leaky buckets accept entire bursts of cells as soon as their parameters allow. In the repetitive burst source model, sources begin the transmission of a burst of cells every $\frac{\text{MBS}}{\text{SCR}}$ units of time. All cells are leaky bucket conformant in this model.

This pattern of cells output from the leaky buckets is shown in Figure 4.4. In this figure, we note that the leaky bucket output follows the pattern of the leaky bucket input.

4.1.2 Leaky Buckets

The leaky buckets that we implemented at each source follow the virtual scheduling algorithm standard of ATM Forum [3].

CBR sources have a single leaky bucket for policing the PCR of the source. This leaky bucket ensures that each CBR source does not transmit cells in excess of its negotiated PCR with the switch. Cells which do not conform to the PCR leaky bucket are dropped and not transmitted to the multiplexer.

VBR sources are more complex and have two leaky buckets for policing the PCR and the SCR, respectively, of the source. A cell must conform to both leaky buckets in order

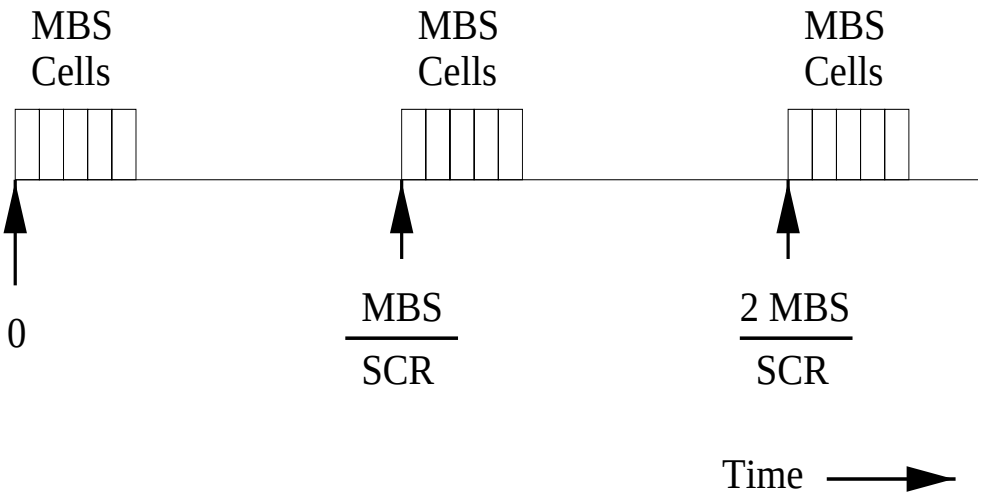

Figure 4.4: Leaky bucket output pattern from a repetitive burst source

for it to be passed from the source to the multiplexer. If a cell conforms to exactly one leaky bucket, then that cell is dropped and the leaky bucket from which it was conformant is reset to unconditionally accept the next arriving cell. If a cell does not conform to either of the leaky buckets, then that cell is dropped and neither of the leaky buckets are reset.

In our simulations, we drop non-conformant cells. In some real-time applications, such as voice or video traffic where a certain degree of cell loss is tolerable, this behavior may be acceptable assuming that the number of conformant cells is large enough to deliver the QoS desired. In most non-real-time applications, such as a file transfer where cell loss is intolerable, this behavior is clearly unacceptable.

In most actual implementations, a source will be able to buffer non-conformant cells at the leaky bucket until such time has elapsed such that they will be conformant. This prevents any cells from being dropped. Naturally, this scenario potentially results in larger delays from a user perspective. Nevertheless, we ignore these delays since our focus is on the delays incurred while a cell is resident in the switch. For a more in depth discussion on user delays versus network delays see p. 15.

4.1.3 Multiplexers

The multiplexer is a place of storage for cells which conformed to their leaky buckets, but need to wait for transmission on their input link due to competition with

cells from other VCs (see Figure 4.1). A multiplexer may not transmit cells at a higher rate than its input link capacity. Also, cells must wait at the multiplexer if another cell is currently being transmitted. The multiplexer consists of a separate, virtually infinite cell queue for each VC that it is multiplexing. If more than one cell is awaiting transmission from a multiplexer, then that multiplexer chooses cells from VCs in a round-robin fashion to ensure fairness.

The amount of delay a cell spends in the multiplexer will add to the delay from a user point of view. For a more in depth discussion on user delays versus network delays see p. 15. The number of VCs that are multiplexed also has a large impact on network observed delay values. For more details see Section 3.2.

4.1.4 Links

A link is simply the physical means on which a cell travels. Our system contains different types of links, including links from a source to the multiplexer, links from the multiplexer to the switch connecting at an input port, and a link from the output port to the destination (see Figure 4.1). Often, the terms link and port are used interchangeably. Here, we mean that a link is the communication path and a port is its interface to the switch.

4.1.5 Buffers

In our system, there are two types of buffers: input buffers and output buffers. Their lengths are given by input parameters. Inside our switch there exists one separate input buffer dedicated to each VC and one common output buffer for all VCs (see Figure 4.1).

Arriving cells are stored in an input buffer until such time that they can be transferred to the output buffer. Since there exists a dedicated input buffer for each VC, each input buffer acts as a FIFO queue for a VC's cells. This FIFO behavior is due to the fact that the deadlines for each cell are calculated using the time of a cell's arrival at the switch (see Equation 3.3).

As long as there are vacancies in the output buffer, cells will continue to be transferred to the output buffer from their input buffer. When a cell arrives at the output buffer, it is inserted into the output buffer in order of increasing deadlines. In the event that the output buffer is full, no cells will be transferred from an input buffer to the output buffer

causing cells to become backlogged in their input buffer. Therefore, in our simulations cells will never be dropped in an attempt to enqueue in the output buffer. Although it never occurred in our simulations, cells could, in theory, be dropped at their input port if their input buffer is full. In this light, we can view the output buffer as merely a shared extension of the input buffers.

As soon as the output buffer is non-empty, the switch will begin transmitting a cell at the output link speed. The switch will continuously transmit cells at this rate until the output buffer becomes empty. While the bits of a cell are being transmitted, we assume that the head of the output buffer is still occupied by this cell. For this reason, we do not allow insertion at the head of the output buffer regardless of the deadline of a cell. If a cell arrives at the output buffer with an earlier deadline than all of the cells in the output buffer, then the arriving cell will be placed in the second buffer slot and transmitted second, assuming no other cells with earlier deadlines arrive before it begins transmission. In the case of ties, a tie is arbitrarily broken in favor of a lower VC number.

Since each VC has its own input buffer and the input buffers are served in a FIFO order, we implemented the input buffers as circular queues. Since we must search the output buffer when enqueueing a cell, we implemented the output buffer using a heap. With Q cells in the output buffer, the heap gave us $O(\log Q)$ search times.

4.1.6 Switch Fabric

Our model of a single ATM switch contains input and output ports, input and output buffers, and a switch fabric. We assume that the switch fabric is non-blocking, such that a cell is never prohibited from being transferred from its input buffer to the output buffer because another cell is transferred to the output buffer at the same time. We simulate this by only allowing one cell to be transferred at a time.

In our simulations we assume that the amount of time required to transfer a cell through the switch fabric with J input ports and a cell transmission time of t_{cell} is uniformly $\frac{t_{cell}}{J}$. We chose this as our cell transfer time so that all newly arriving cells can be transferred to the output buffer before the next cells arrive at each port.

4.2 Parameter Set and Values Used in Experimentation

In this section we describe the parameters that are input to the simulator and the values that were frequently chosen in our experiments. A discussion of the impact of the variation of these parameters follows. For reference, an example input file is given in Appendix A.

4.2.1 Number of Cells

The number of cells input parameter dictates how many total cells, regardless of VC, pass through the switch. Once this number of cells has exited the switch, the simulator stops and outputs various statistics. In order to get more accurate results, our goal has been to pass as many cells through the simulator as possible. Often, time dictated how many cells each simulation produced. Most of our experiments were run for 10,000,000 total cells. At a minimum, experiments were run for 1,000,000 total cells. These numbers were chosen arbitrarily and proved to have a great impact on the run-time of the simulations.

4.2.2 Number of Input Ports

The number of input ports input parameter tells the simulator about the topology of the switch. The simulator dynamically creates this number of input ports onto which VCs will be multiplexed. The values chosen for this parameter were 1, 2, 4, 8, 16, 32, and 64. These values were chosen so that we could observe the effects of the degree of multiplexing (see Section 3.2) by varying the number of input ports and keeping the total number of VCs constant.

4.2.3 VC Topology

The VC topology input parameter has two purposes. First, it defines different classes of VCs in the simulation by enumerating them starting with 1. Secondly, it defines how many VCs are multiplexed onto each input link. We experimented with both single and multiple VC classes.

4.2.4 Size of Input Buffers

The size of input buffers input parameter dictates the cell capacity of every input buffer in the switch. Each VC has its own dedicated input buffer. All experiments were performed with each input buffer able to store 5,000 cells. This value, coupled with the size of the output buffer, was chosen large enough so that there would not be any cells dropped inside of the switch due to limited buffer capacity.

Further justification of our choice of input buffer size is necessary. In Equations 2.1 and 2.2, Georgiadis *et al.* [7] give a theoretical lower and upper bound range for our fixed allocation input buffers. In our discussion in Section 2.1, we stated that these theoretical bounds were overly conservative. As an example, in a typical experiment set we simulated 256 VCs with an MBS of 171 cells. Using these values with Equations 2.1 and 2.2, the lower and upper bounds for our input buffer capacity are 44,288 cells and 88,065 cells!

In the interest of executable file sizes, we implemented our input buffers with capacities of 5,000 cells and closely monitored the input buffer behavior. We found this value to be overly sufficient as cells were never dropped due to lack of input buffer space. The input buffers rarely contained more than 100 cells at any given time.

4.2.5 Size of Output Buffer

The size of output buffer input parameter dictates the cell capacity of the single output buffer in the switch. Each VC shares a common output buffer. All experiments were performed with the output buffer able to store 5,000 cells. This value, coupled with the size of the input buffer, was chosen large enough so that there would not be any cells dropped inside of the switch due to limited buffer capacity. During our simulations, cells were never dropped inside of the switch. During our simulations, the output buffer was often full at which point cells were backlogged into their input buffer.

We note that since the output buffer is merely a shared virtual extension of the input buffer space, we do not need to give further consideration to its theoretical buffer size bounds.

4.2.6 Input and Output Link Speed

The input and output link speed input parameter determines the transmission rate of cells into and out of the switch. The simulator requires that these rates be the same;

however, the transmission speeds of the sources may vary. In all of our experiments, we set this value to 149.76 Mbps[1].

4.2.7 Source Speeds

The source speeds input parameter determines the transmission rate of cells from a source, through its leaky buckets, to the multiplexer. The source speeds can vary for each class of VC. In our experiments we did not vary this value, setting it to 149.76 Mbps[1] in all experiments.

4.2.8 Packet Size

The packet size input parameter determines the number of bytes of data that are packetized for transmission, as explained in Section 4.1.1. This parameter may be varied for each class of VC. For most experiments, we set this value to 8 KB for all VCs in order to simulate Transmission Control Protocol (TCP) packets. This parameter is relevant for VBR traffic and not for CBR.

4.2.9 Idle Period Mean

The idle period mean input parameter determines the amount of time (in μsec) that a VBR source, on average, will remain in the idle state between successive active periods, as explained in Section 4.1.1. This parameter is relevant for VBR traffic and not for CBR. In our experiments the mean idle periods were varied to create different levels of total bandwidth reservation. Typical values included 100,000 μsec, 300,000 μsec, and 500,000 μsec.

4.2.10 Pause Period Mean

The pause period mean input parameter determines the amount of time (in μsec) that a VBR source, on average, will remain in the pause state between successive packet transmissions, as explained in Section 4.1.1. This parameter is relevant for VBR traffic and not for CBR. In our experiments the mean pause periods were varied greatly to achieve different levels of total bandwidth reservation. Typical values included 125,000 μsec,

[1] Commonly referred to as 155 Mbps, 149.76 Mbps is the OC-3 data transmission rate as described in the ATM Forum [21].

135,500 μsec, 156,400 μsec and 182,000 μsec. These values were chosen through trial and error methods in order to determine the exact amount of bandwidth reservation desired.

4.2.11 Number of Packets per Active Period Mean

The number of packets per active period mean input parameter determines the number of packets that a VBR source, on average, will generate while the source is in the active state. Therefore, this parameter, along with the pause period mean parameter, determines how long a VBR source is in the active state, as explained in Section 4.1.1. This parameter is relevant for VBR traffic and not for CBR. In our experiments the mean number of packets per active period were varied to create different levels of total bandwidth reservation. Typical values included 50 and 100 packets.

4.2.12 Source Start Times

The source start times input parameter determines the time at which a source starts transmitting its first cell. This parameter can be varied for each VC class in the system. For our experiments, the source start time was assigned to 0 for all VCs so that we could focus our efforts on studying the simulator's warm-up period.

4.2.13 Traffic Mode

The traffic mode input parameter tells the simulator which VC classes are CBR traffic and which are VBR traffic. If a numerical value is specified for this parameter, then the simulator will assume that this is a CBR VC. The CBR VC's PCR will be set to the inverse of this input value in μsec. If a word is entered that begins with a "v" or "V", then the simulator assumes that this a VBR VC and uses the input parameters: packet size, mean idle period, mean pause period, and mean number of packets to determine the VC's SCR. For a more detailed description on how this value is calculated see Section 4.2.18. In most of our experiments we used the VBR setting for this input parameter.

4.2.14 Cell Delay Variation Tolerance for Peak Cell Rate

The CDVT_{PCR} input parameter is used in determining the limit value for the PCR leaky bucket. This value is given in μsec. In all of our experiments, the CDVT_{PCR} value was set to zero.

4.2.15 Cell Delay Variation Tolerance for Sustained Cell Rate

The $CDVT_{SCR}$ input parameter, along with the BT, determines the limit value for the SCR leaky bucket. This value is given in μsec. In most of our experiments, the $CDVT_{SCR}$ value was set to zero. In our experiments using the repetitive burst source model, we set the $CDVT_{SCR}$ to 1 μsec in order to offset precision errors. We observed that although each cell should, in theory, be conformant to the SCR leaky bucket, the last cell of each burst was being dropped due to precision errors in our simulator. By setting the $CDVT_{SCR}$ to 1 μsec this last cell became conformant.

4.2.16 Verbose Modes

Several verbose modes exist in the simulator for debugging purposes. These verbose modes allow us to print out debugging statements to a file and to the terminal in regards to several different aspects of the simulator including: the sources, the leaky buckets, the event list, and the motion of cells through the switch. These verbose modes can be set on in any combination by specifying a word beginning with "t" or "T" for true. These verbose modes do not affect the results of the simulation, but they do drastically increase run-time and file sizes.

4.2.17 Random Number Seed

The random number seed input parameter is used to provide the initial seed to the stream of random numbers generated by the `erand48()`function used by the simulation. This parameter was arbitrarily set to zero for all of our experiments.

4.2.18 Derived Values

The following parameters are not explicitly input into the simulator, but are instead derived from the above input values.

The MBS is set equal to the size of a packet in cells. The packet size is given in bytes, so the MBS is equal to the ceiling function of the packet size divided by 48, since there are 48 bytes in an ATM cell's payload. The ceiling function is used since we pad the last cell with zeros if necessary.

In the CBR case, the PCR is equal to the inverse of the input parameter described in Section 4.2.13. In the VBR case, the PCR is assigned as the amount of time to transmit

one cell at the source's link speed.

In the CBR case, the SCR is equal to the PCR. However, in the VBR case the SCR is dictated by the three state model parameters, as described in Section 4.1.1, where $\overline{M}$ is the mean number of packets per active period, t_{packet} is the amount of time required to transmit a packet, $\overline{I}$ is the mean idle period, $\overline{P}$ is the mean pause period, and S is the packet size in bytes. Then the SCR is given by the following formula,

$$\text{SCR} = \frac{\overline{M}\left\lceil \frac{S}{48} \right\rceil}{\overline{M}t_{packet} + \overline{I} + (\overline{M} - 1)\overline{P}}. \tag{4.2}$$

Given that t_{cell} represents the amount of time required to transmit a cell at the speed of the output link, the theoretical maximum delay bound is computed according to the formula,

$$d^{max} = \frac{\text{MBS}}{\text{SCR}} + t_{cell}. \tag{4.3}$$

This formula is analogous to Equation 3.2. More detail on theoretical maximum delay bounds can be found in Section 3.5.

The BT is used in the SCR leaky bucket. We used the ATM Forum's recommendation [3] in calculating the BT as follows:

$$\text{BT} = (\text{MBS} - 1)\left(\frac{1}{\text{SCR}} - \frac{1}{\text{PCR}}\right). \tag{4.4}$$

4.3 Parameter Prioritization and Impact

After designing the simulator to be as flexible as possible, one of the challenges we are left with is in choosing our parameter sets for experimentation. Clearly, from the above discussions our parameter space is infinite. In this section we justify our choices above.

Among the parameters most often varied was the number of input ports. By keeping the total number of VCs constant and varying the number of input links we vary the degree of multiplexing. This has a very profound impact on the delay values we obtain. The greater the degree of multiplexing, the less worst-case delays we observe. For a more in depth discussion regarding the degree of multiplexing see Section 3.2.

As can be seen in Equation 4.2, the choice of the mean pause period, $\overline{P}$, has the most profound impact on the resulting SCR value, since $\overline{P}$ dominates the denominator given a large mean number of packets per active period, M and a small t_{packet}. The idle period

mean and number of cells mean were also often varied, but these parameters do not have nearly as much impact on the SCR and therefore the reservation level (see Equation 3.1) as does the mean pause period, $\overline{P}$.

Perhaps the most critical of the parameter variations for our studies is the number of VC classes we simulate. In the homogeneous cases, we observe the tails of the cell delay functions to be monotonically decreasing. However, in the heterogeneous cases, we note that the competition between the various VC classes can lead to peaks at the tails of the cell delay functions.

The remaining parameters not mentioned here were not deemed as important for the purposes of our experimentation. These parameters were rarely, if ever, varied. However, the varying of these parameters could lead to interesting future work.

4.4 Bin Size Rationale

In order to produce meaningful delay measurements from our simulations, we must keep track of the amount of delay each cell has incurred upon exiting the switch. As the number of total cells increases, this becomes an issue both in program size and output size. Clearly, we would not have enough memory to store 10,000,000 real numbers for the entire duration of our simulation. Even if we could, managing all of these values would not be practical. Therefore, we introduce the concept of cell delay "bins".

Each VC in the simulator has allocated to it a pre-defined number of cell delay bins. As each cell exits the switch, the bin whose delay range covers the actual delay value gets incremented. The width of these bins is determined as a compile-time parameter. Each VC is allocated enough bins such that the sum of the bin widths for each VC is at least as large as its theoretical maximum delay bound.

In our initial simulations, the bin width was set to the time it takes for one cell to be transmitted (2.8312 μsec at 155 Mbps). As we increased the number of VCs to 256 and also increased the theoretical maximum delay bounds, it soon became apparent that the number of bins was making the program and the output too large. Thus, we increased the width of the bins to the time it takes for 10 cells to be transmitted (28.312 μsec at 155 Mbps). Even this increase in coarseness was insufficient for obtaining a reasonably sized executable file and output files, so the bin width was finally increased to the time it takes for 100 cells to be transmitted (283.12 μsec at 155 Mbps). Each of our experiments

reported here was run with the bin width equal to 100 cell transmission times.

A disadvantage of increasing the bin width is that we lose the detail of knowing exactly how many cells exited the switch with a given amount of delay on a per-VC basis. However, this lack of precision is acceptable since we are examining comparatively large theoretical maximum delay bounds.

All cell delays are reported as the value of the right-most (greatest) delay in the bin range. Therefore, we are reporting the worst-case behavior as each cell in a bin may have had a delay anywhere in that bin's delay range.

4.5 Worst-Case Cell Delay and The Warm-Up Period

We focus our research on worst-case cell delays and thus the tails of cell delay distribution functions. Our interest in worst-case cell delay is coupled with our interest in EDF's behavior in these worst-case scenarios. Through our experimental results, we provide a barometer for both future research in this field and also in ATM switch development using EDF scheduling.

An advantage of our worst-case approach is that we are able to offer delay guarantees independent of the actual network traffic. The disadvantage of this approach is that while these worst-case scenarios are possible, they would rarely occur in an actual ATM network. It should therefore be kept in mind that the results from our worst-case experiments are more pessimistic than would likely be observed under average network conditions.

To simulate worst-case scenarios, in most of our experiments we aligned each source to start transmitting cells at the same instant of time through the use of the source start times parameter. When all sources transmit bursts of cells simultaneously at the beginning of the simulation, worst-case delays result since the switch is flooded with cells. We use the term "warm-up period" to denote that time from simulation onset until a switch has recovered from the initial flood of cells in the aligned sources case. After its warm-up period, the switch begins serving cells with much less delay. There is no exact measure of the warm-up period as warm-up periods vary greatly depending on the switch topology, initial burst sizes, and subsequent burst sizes.

Finally, in analyzing our simulation results, we focus on the VC from each class for which its maximum delayed cell is also the maximum delayed cell over all VCs in its class.

Chapter 5

Simulation Results

In the course of our research we have studied worst-case cell delays inside of an ATM switch using EDF scheduling. In our work we experimented with four different source models: the three state model, the persistent model, the repetitive burst model, and the MPEG trace model.

The three state model is described in detail in Section 4.1.1. This model generated very bursty VBR traffic and was chosen for experimentation largely because it is recommended by the ATM Forum in [3].

Using the three state model, we studied the effect of the degree of multiplexing for VCs belonging to a common VC class. We observed that as fewer VCs are multiplexed onto each link, the cell delays inside of the switch approach their theoretical maximum delay bounds during the switch's warm-up period. We note that after the switch's warm-up period very minimal cell delays are observed, regardless of the degree of multiplexing. We observed that in these experiments the cell delay functions decrease monotonically and then alternate between one and zero cells at the tails of these functions.

We also studied delay bound percentages for a variety of different degrees of multiplexing and output link reservation levels. In these experiments, larger cell delays occur as the degree of multiplexing decreases and also as the output link reservation level increases. We observed substantial decreases in delay bound percentages from a minimal drop in delay guarantee percentage. However, we note that since the worst-case cell delays occur during the switch's warm-up period, it is possible to provide arbitrarily favorable delay bound percentages by increasing the length of the simulation.

Also using the three state model, we studied the effects of multiple classes of VBR

traffic. We examined the characteristics of the second burst of cells from a particular VC and the effect on its delay due to cells from other VCs. We observed that the second burst of cells can cause a sharp peak at the tails of delay functions. Through manipulation of the three state source model we determined that we could decrease delays incurred by the second burst of cells.

In another experiment using multiple classes of VBR traffic, we observed that delays in consecutive bursts of cells can increase during the switch's warm-up period. Through manipulation of the three state source model we determined that we could maximize the delays incurred by any single burst of cells.

Also using the three state model, we studied the behavior of the EDF scheduler by focusing on the sequence in which cells are output from the switch on a VC class basis. We observed that as the degree of multiplexing decreases, cells from initial bursts tend to be transmitted back-to-back with other cells in their same VC class.

The persistent source model and the repetitive burst source model were designed to examine the worst-case behavior of the leaky buckets at each source. These two patterns of source behavior explore the limits of the leaky buckets in terms of accepting individual cells and accepting bursts of cells, respectively.

In the persistent model, a source continuously sends cells to its leaky buckets. The leaky buckets are such that they accept the first burst of MBS cells and after which they accept individual cells at a rate equal to the SCR. We chose to study this source model because of our interest in examining cell delays under worst-case leaky bucket behavior. This source model produces worst-case leaky bucket behavior in the sense that the leaky buckets always accept a cell as soon as the leaky bucket parameters allow.

With the persistent source model, we studied multiple classes of VCs and aligned all sources to begin transmission simultaneously as well as staggered each source to start transmission at a random point in time. Our results showed that in the worst-case alignment of sources, cell delays during the switch's warm-up period were very large. Further, we observed that the cell delay functions were roughly constant during the switch's warm-up period. We also observed a drastic decrease in cell delays when staggering the sources' initial transmission times.

In the repetitive burst model, a source repeats a cycle of sending a burst of MBS cells to its leaky buckets and then waiting before sending another burst of MBS cells. The wait time between each burst is just long enough such that every cell of the burst will

be conformant to the leaky buckets. We chose to study this source model because of our interest in examining cell delays under worst-case leaky bucket behavior. This source model produces worst-case leaky bucket behavior in the sense that the leaky buckets always accept an entire burst of cells as soon as the leaky bucket parameters allow.

With the repetitive burst model, we studied multiple classes of VCs in both the aligned and staggered mode of initial cell transmission. We observed that in the aligned mode the switch has an infinite warm-up period in the sense that it is not able to serve the vast majority of its cells with very minimal delays. This is not the case with each of our other source models. We also observed that in the aligned mode, cell delays closely approach their theoretical maximum delay bounds and that the tail of the cell delay function decreases concavely. The worst-case cell delays in this experiment are slightly smaller than in the analogous persistent source model experiment. Finally, we observed major decreases in cell delays when sources were initially staggered; however, these delay improvements were not as profound as in the persistent model case.

In the MPEG trace model, actual MPEG video traces were used to determine source cell generation times. We chose to examine this source model in order to compare actual video traffic cell delays to cell delays incurred in our other worst-case models.

In our experimentation using MPEG video traces, we simulated worst-case MPEG behavior. In the experiment reported here, we aligned all sources to transmit the first cell of each frame simultaneously with subsequent cells in each frame transmitted consecutively. We observed that the tails of the delay functions are much further from their theoretical maximum delay bounds than any of our other worst-case experiments.

In addition to the aforementioned VBR experiments, we also experimented with CBR sources; however, the deterministic nature of CBR produced very uninteresting results and they are therefore not reported here.

Details of our simulation results are organized in this chapter according to their source behavior. We begin our discussion in Section 5.1 with a study of both the homogeneous and heterogeneous cases of the three state model. In Section 5.2, we explore the aligned and staggered cases of the persistent source model. In Section 5.3, we examine the aligned and staggered cases of the repetitive burst source model. Finally, in Section 5.4, we show the results from an MPEG source trace experiment.

Appendix C is provided as a reference of the parameters which were varied in the experiments presented in this chapter. Appendix C also lists output link utilization

percentages observed in our experiments.

5.1 Three State Source Model Experiments

Each of the experiments discussed in this section use the three state source model for VBR traffic as described in detail in Section 4.1.1.

5.1.1 Homogeneous Virtual Circuits

In each of these experiments, we simulated one class of VC traffic. We refer to these VCs as "homogeneous" or of the same VC class since each VC is given an identical set of input parameters, such as PCR, SCR, MBS, etc.. Note that although VCs are homogenous, the randomness introduced by the three state model prevents each source from transmitting cells in an identical fashion.

The Effect of the Degree of Multiplexing

In the following two experiments we kept all input parameters constant—only varying the number of VCs multiplexed onto each link. Our intent was to examine the impact of the degree of multiplexing. As more VCs are multiplexed onto an input link, the bulk of delay that a cell encounters during the switch's warm-up period tends to shift from occurring inside the switch to occurring inside the multiplexer. The inverse of this is true as the degree of multiplexing is decreased. From a user's perspective, where the delay occurs along a cell's path is irrelevant. However, our focus on delays is from the perspective of the switch which is only concerned with cell delays incurred while it contains the cells. It is to the benefit of the user if a large degree of multiplexing can be achieved, since this will allow the switch to accept more VCs while still providing the same end-to-end QoS to the user at a reduced cost.

Each of these two experiments was run for 10,000,000 total cells and consisted of 256 homogeneous VBR VCs which each used the three state model. The input parameters of the three state model (mean idle period, mean pause period, packet size, and mean number of packets per active period) were deliberately chosen such that the percentage of output link bandwidth reserved for these connections would be approximately 97%. This reservation level was chosen to be relatively close to the output link capacity because of

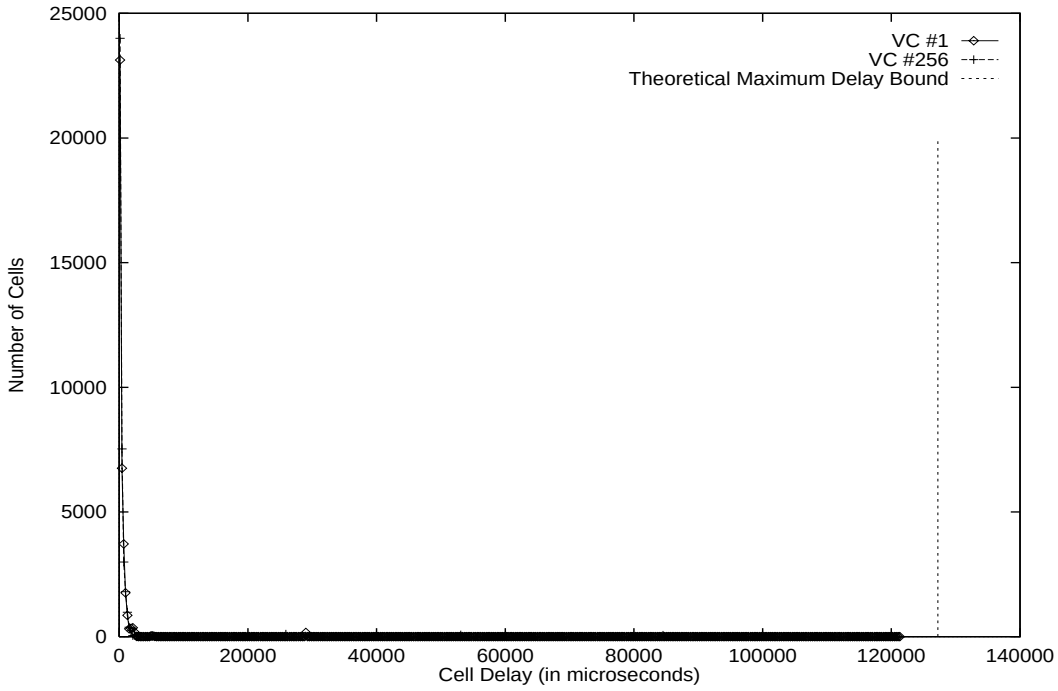

Figure 5.1: Cell delays with 256 VCs on 64 input links

our interest in examining worst-case behavior. Intuitively, the larger the reservation level of an output link, the greater the potential for competition among cells using the link, thus a better chance for larger cell delay values. For more information on how the CAC algorithm calculates the amount of output link bandwidth to reserve on a per VC basis see Section 3.1.

The first of these two experiments, shown in Figure 5.1, consisted of 64 input links resulting in 4 VCs per input link. The second experiment, shown in Figure 5.2, consisted of 2 input links resulting in 128 VCs per link. Due to the large number of VCs present in these simulations, we plot only the VCs with the worst and best observed maximum delay values. These worst-case and best-case VCs are VC #256 and VC #1 in Figure 5.1, respectively and VC #134 and VC #1 in Figure 5.2, respectively.

In both Figures 5.1 and 5.2, we observe that the vast majority of cells for these VCs incur very small delays, as evidenced by the large peaks at the left-most edge of the delay functions. We also observe that after these large peaks, the delay functions decrease dramatically after which a long tail is produced. Cells which contribute to the tails of these functions are those which depart during the switch's warm-up period. Since in these experiments all sources are aligned to begin transmitting simultaneously, the switch is

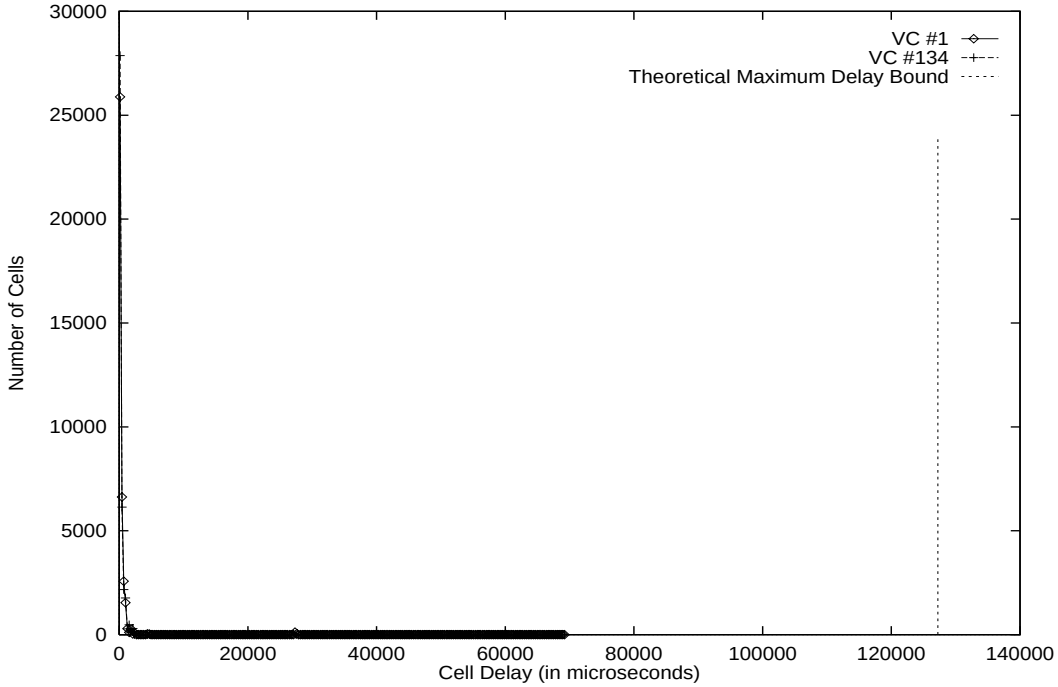

Figure 5.2: Cell delays with 256 VCs on 2 input links

flooded with cells from all sources and large delays result. After this warm-up period, source transmissions are more dispersed in time, thus these cells will encounter less competition with other cells inside of the switch and smaller delays result.

The theoretical maximum delay bound for all cells is marked on these figures with a vertical line at 127,237 μsec. This value was calculated according to Equation 4.3. We note that due to the homogeneous nature of all of the VCs in these experiments, their theoretical maximum delay bounds are all equivalent. As discussed in Section 3.5, the theoretical maximum delay bounds are an upper limit to permissible cell delay in order for real-time traffic to meet its playback instant. We observe that because we did not overallocate the output link, all cells are delivered by their theoretical maximum delay bound.

In comparing these two experiments, we note that in Figure 5.1 the delay "gap" between the maximum cell delays incurred and the theoretical maximum delay bound is relatively small during the warm-up period, whereas in Figure 5.2 this delay "gap" is much larger. These results are evidence of the degree of multiplexing property discussed in Section 3.2. In Figure 5.1, during the warm-up period cells spend less time competing with

other cells in the multiplexer, yet spend more time competing with other cells once inside the switch. Since we measure our cell delays as the difference in time between cell arrival at the switch and cell departure from the switch, larger delays are observed. For Figure 5.2, the opposite is true.

It is important to note that in our homogeneous VC experiments, we did not observe the effects of EDF scheduling. Since all VCs had the same input parameters, the calculated theoretical maximum delay bound was constant for all VCs, thus causing deadlines to increase uniformly as arrival times increased. Therefore, cells were served in the order of their arrivals and our EDF scheduler behaved as if it was a FIFO scheduler (see Section 1.2.2).

In Figure 5.3, we zoom in on the tail of the cell delay function presented in Figure 5.1. In Figure 5.3, we show the four VCs which are multiplexed onto the first input link into the switch. At the tails of these functions we observed an alternating pattern of one or zero cells falling into each cell delay bin. These patterns are due to the fact that the initial burst of cells for these VCs are served in a round-robin fashion. That some cells fall into the same cell delay bin is attributed to our setting the delay bin width to 100 times the unit of cell transmission in these experiments.

Percentages of Delay Bounds

Next, we examine percentile delay guarantees as described in Section 3.1. In our percentile delay guarantee experiments we simulated 28 different scenarios varying only the degree of multiplexing from 256 to 4 VCs per link and the amount of output link bandwidth reserved from a 97% to a 65% reservation level. For each of these experiments we simulated 10,000,000 total cells on 256 total VCs. Also, VCs were distributed uniformly over the input links and each VC used the same input parameters in generating VBR traffic according to the three state model. (Note that Figures 5.1 and 5.2 are specific examples from this experiment set.)

An explanation of our method of calculating percentile guarantees is necessary. Because we use cell delay bins (see Section 4.4) as our unit of measuring cell delay, we are restricted to this level of granularity. Therefore, on a per-VC basis, we declare that an $n\%$ delay guarantee is first satisfied at time t, where t is the right edge of the first cell bin in which at least $n\%$ of the total number of cells have fallen into it and all earlier cell bins

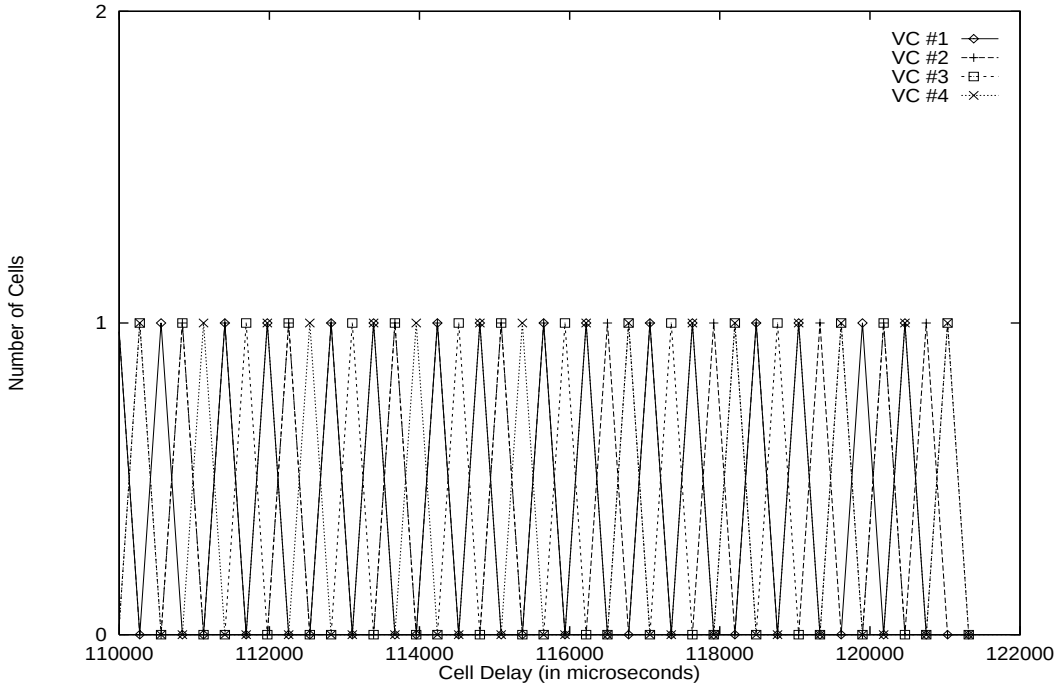

Figure 5.3: Tail behavior of cell delays with 256 VCs on 64 input links

cumulatively.

Graphs of the 97% and 65% reservation levels are given in Figure 5.4 and Figure 5.5, respectively. In Figures 5.4 and 5.5, we plot 5 different delay guarantee percentiles ranging from 100% to 95% as a percentage of the common theoretical maximum delay bound for 7 different degrees of multiplexing ranging from 256 VCs per input link to 4 VCs per input link. Since the input parameters were the same for each VC, the calculated theoretical maximum delay bounds are therefore the same for each VC.

In Figure 5.4, we observe that with a low degree of multiplexing, providing a 100% or 99.99% delay guarantee results in delays which closely approach the theoretical maximum delay bound. Note that for a degree of multiplexing equal to 256 there is only 1 input link into the switch; therefore, each cell is immediately served in a FIFO order and little cell delay is incurred inside of the switch.

Also of interest is the large drop in cell delay (dropping in half for low degrees of multiplexing) resulting in the small drop from a 99.99% guarantee to a 99.5% guarantee. This is attributed to the fact that with a low degree of multiplexing, only a small percentage of cells incur very large delays. These large delays occur during the simulator's warm-up

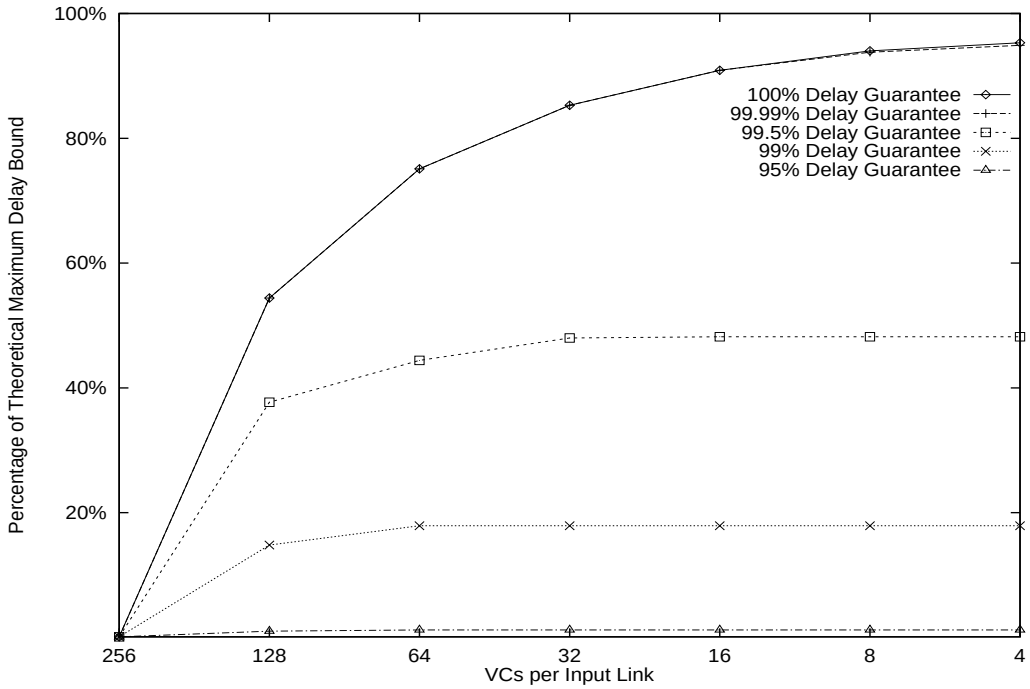

Figure 5.4: Delay percentages with 256 VCs and 97% output link reservation

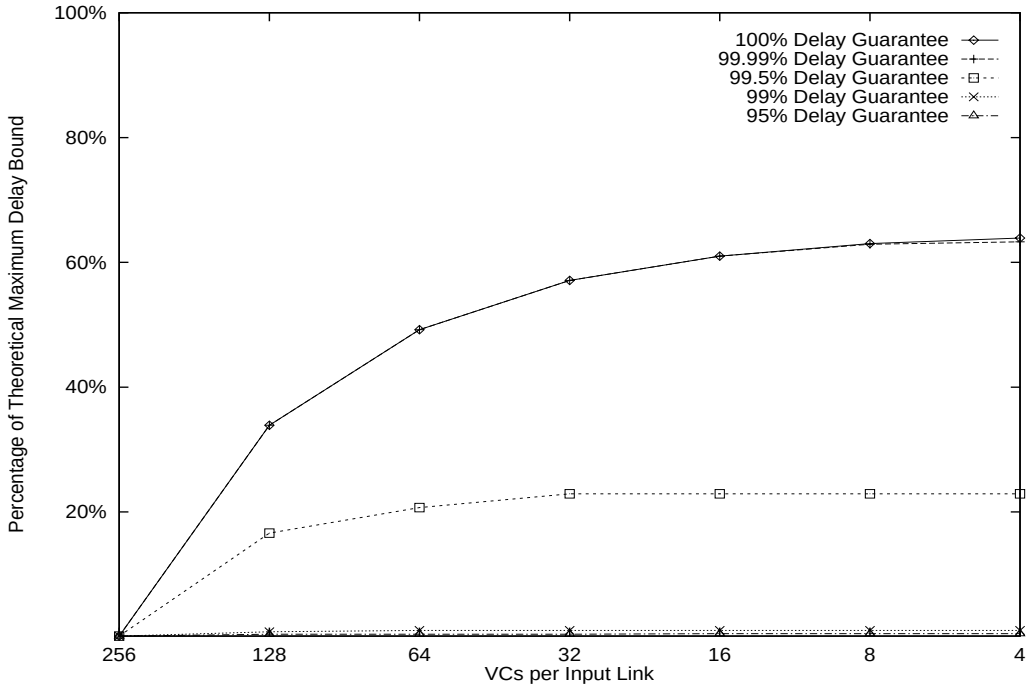

Figure 5.5: Delay percentages with 256 VCs and 65% output link reservation

period since all sources are synchronized to start transmitting at the same instant. In Figure 5.4 it is also interesting to note that 95% of the cells encounter very minimal cell delays regardless of the degree of multiplexing.

By comparison, Figure 5.5 shows even more favorable delay guarantee results. This is intuitive since the SCR in this case is smaller for each VC than in Figure 5.4. In Figure 5.5, 99% of the cells encounter very minimal cell delays.

A very important point must be explained when analyzing these percentile guarantee results. In each of the experiments described here, 10,000,000 total cells were passed through the switch. We have observed that because the largest cell delays occur during the warm-up period, an increase in the total number of cells simulated minimizes the effect of the warm-up period cell delays on the entire simulation. Therefore, by increasing the number of cells simulated, we can provide arbitrarily favorable delay guarantees. For this reason, we change our focus from worst-case percentile guarantees to worst-case heterogeneous VCs.

5.1.2 Heterogeneous Virtual Circuits

Up until this point we have shown experiments where all of the VCs were homogeneous. As a result of this, our EDF scheduler behaved as a FIFO scheduler would. Next we examine heterogeneous VC behavior.

In these experiments we define two separate VC classes (roughly simulating 2 classes of VBR traffic, such as two types of video; one requiring more bandwidth than the other). We refer to these VC classes by assigning them letters in decreasing alphabetic order of their SCRs. For example, a class A VC would have a greater SCR than a class B VC. All VCs within a class are given the same three state model parameters. In these experiments we observe the EDF nature of our scheduler.

Second Burst Behavior

In the next two experiments, we analyze what we call the "second burst behavior" of a VC. In the first experiment, we simulated 10,000,000 total cells from 256 VCs multiplexed uniformly onto 32 input links. For each VC, the MBS was set to 8 KB, or 171 cells. The three state model parameters for each VC class were chosen such that the output link reservation level was set at approximately 97% of its 155 Mbps capacity and the ratio of class A SCRs to class B SCRs was 1.85:1.

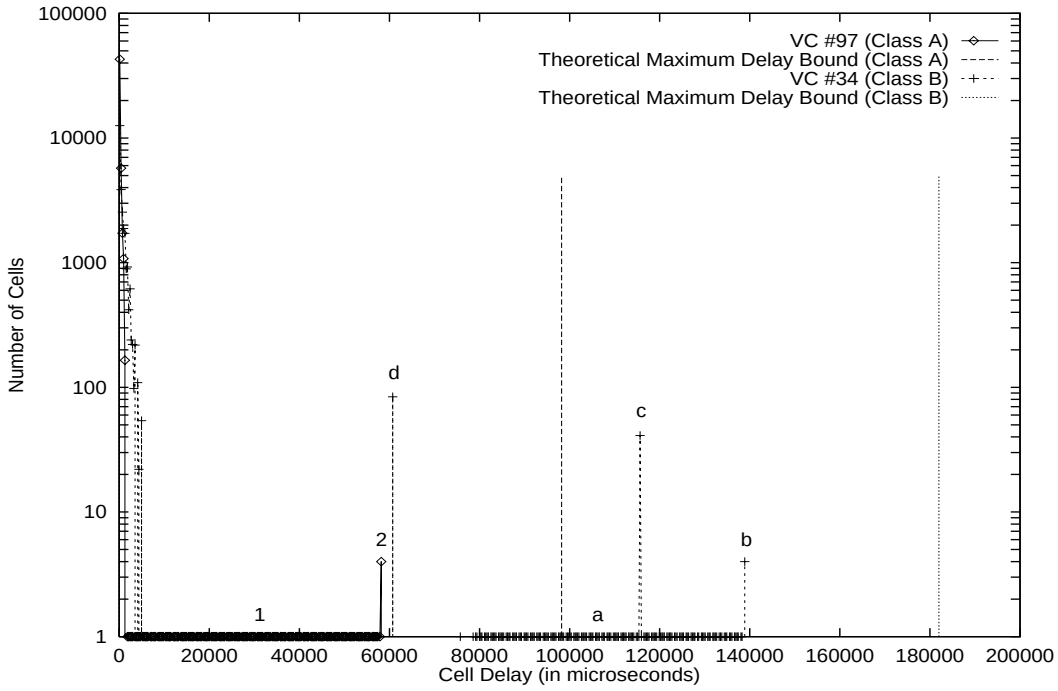

Figure 5.6: Cell delays with 256 VCs in 2 VC classes on 32 input links

The cell delay function is shown in Figure 5.6. Due to the large number of VCs simulated, we plot only the VCs in each class which have the greatest cell delay. These are VC #97 for class A traffic and VC #34 for class B traffic.

VC #97's first and second bursts are marked with a "1" and "2", respectively. VC #97's first burst of cells is represented in the range of delay bins from approximately 1,000 μsec to 58,000 μsec and produces a height of one cell in each delay bin. VC #97's second burst of cells creates the small peak in the delay function at approximately 58,000 μsec.

Similar to VC #97's first burst, VC #34's first burst, labeled "a" in Figure 5.6, spans the cell delay bins from approximately 75,000 μsec to 139,000 μsec. VC #34's next 3 bursts, labeled "b", "c", and "d", respectively, create the peaks at approximately 139,000 μsec, 115,000 μsec, and 60,000 μsec.

All subsequent bursts for both VC #97 and VC #34 occur after the switch's warm-up period and therefore incur very small delays as evidenced by the extremely large peaks at the left-most edge of the delay functions in Figure 5.6.

It is interesting to note that by aligning the transmissions of the first burst of cells, the first bursts of cells were served in a round-robin sequence within each VC class. This

results in large ranges of bins for these bursts similar to those observed in Figures 5.1 and 5.2.

The fact that VC #34 and VC #97 had the largest cell delays in their respective classes is attributed to the randomness of the three state model which gave these VCs small first pause periods. Since the BT (see Equation 4.4) is used in the limit parameter of the leaky bucket, the first burst of cells always conforms to the leaky buckets. With these small first pause periods, the second burst from VC #34 and from VC #97 arrive at their leaky buckets relatively soon after the first burst has left the leaky bucket. Because of this, only a few of the second burst of cells are conformant to the SCR leaky bucket and the rest are dropped. Because these leaky bucket conformant second burst cells arrive at the switch after all of the first burst cells, their deadlines are such that they must wait until all first burst cells from their entire VC class have been served. Once all of the first burst cells in their VC class have exited the switch, the second burst cells from VC #34 and VC #97 must only compete with the few other VCs with similarly small first pause periods. Because of this, the second burst of cells for these VCs exited the switch nearly consecutively and therefore were placed into the same delay bin, resulting in a peak in the cell delay function.

Clearly, the large cell delays incurred by the first four bursts from VC #34 are attributed to their competition with both class A and class B cells. Since in this experiment all sources are aligned to begin transmission at the start of the simulation, class B cells are buffered in the switch until such time elapses that they are closer to their theoretical maximum delay bound than any other cell. Here we see EDF in action! Intuitively, during the warm-up period, class B cells have larger delay values than class A cells, since class B cells have a smaller SCR and thus a larger theoretical maximum delay bound.

In a previous experiment (see Figure 5.3), the tails of the cell delay functions alternated between one and zero cells per bin. However, this is not the case in Figure 5.6 as the second burst caused a peak to occur at the tail of the distribution function. This is somewhat alarming if percentile guarantees are desired, since naturally a peak at the tail of the distribution function will result in a worse x^{th} percentile guarantee for all $x < 100\%$.

From this experiment we devised a series of controlled experiments to study the behavior of the second burst of cells from VC #97. By manually setting the first pause period for VC #97 and keeping all other input parameters constant, we were able to decrease, but not increase, the delay of the second burst of cells from VC #97.

We were not able to increase the delay of the second burst since manually de-

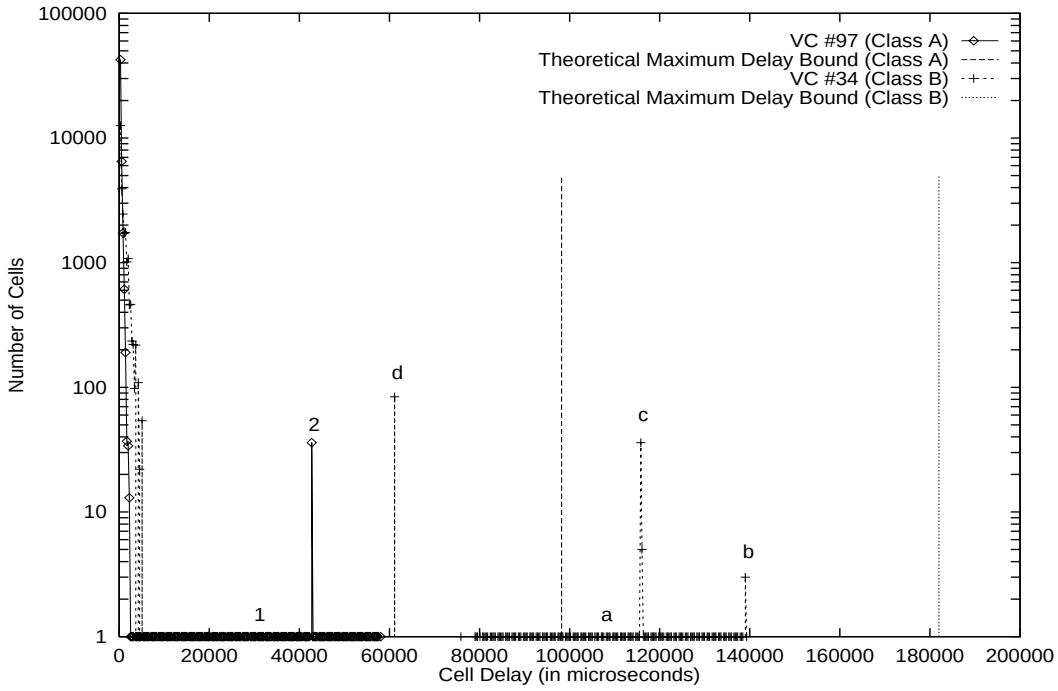

Figure 5.7: Manipulation of the second burst of cells

creasing the already small first pause period for VC #97 merely caused more cells to be non-conformant at the leaky buckets. This resulted in a smaller second burst size for VC #97 and therefore less cell competition for those second burst cells. We predict that manually decreasing the first pause periods for several other class A VCs would cause greater cell delays in the second burst of VC #97.

We were, however, able to decrease the cell delay of the second burst cells of VC #97 by manually increasing VC #97's first pause period as shown in Figure 5.7.

In Figure 5.7, we show a controlled experiment where VC #97's first pause period was manually set to 20,000 μsec. In increasing this pause period, we observed that VC #97's second burst of cells encountered less competition from class A first burst cells. Therefore, the delays incurred by VC #97's second burst are reduced. By increasing this pause period we also observed that more cells from VC #97's second burst were conformant to the leaky buckets, since enough time had elapsed such that more of these second burst cells arrived within the limit parameter of the SCR leaky bucket. This resulted in an increase in the size of the peak at approximately 43,000 μsec.

The cell delay and number of cells in the peak attributed to the second burst in

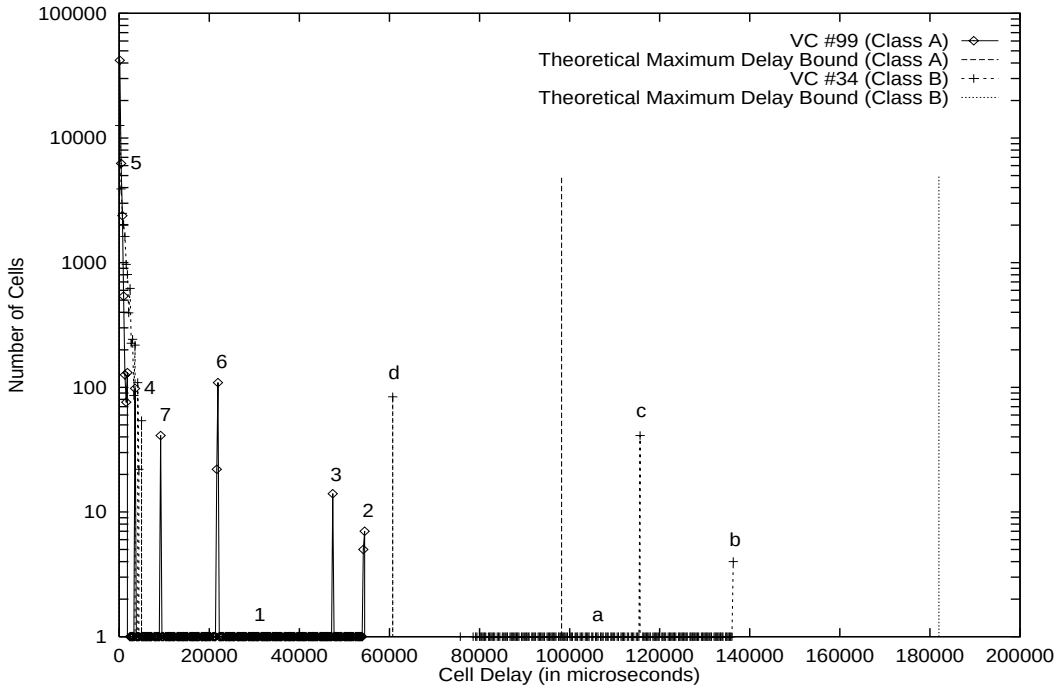

Figure 5.8: Cell delays with 256 VCs in 2 VC classes on 16 input links

this figure can be compared to Figure 5.6, where the first pause period for VC #97 was randomly chosen according to the exponential distribution as 1,921.14 μsec.

Second Bounce Behavior

In another experiment set, we analyze what we call the "second bounce behavior" of a VC. In the first such experiment, we simulated 10,000,000 total cells from 256 VCs multiplexed uniformly onto 16 input links. For each VC, the MBS was set to 8 KB, or 171 cells. The three state model parameters for each VC class were chosen such that the output link reservation level was set at approximately 97% of its 155 Mbps capacity and the ratio of class A SCRs to class B SCRs was 1.85:1.

Again, we plot only those VCs, VC #99 and VC #34, with the greatest amount of cell delay in their respective VC class. VC #99 and VC #34 had the greatest amount of delay because their first pause periods are relatively short and thus have peaks at the tails of their distribution functions. This graph is shown in Figure 5.8. In this figure, the first 7 bursts from VC #99 are numbered consecutively on the graph.

The most striking result from this experiment is in the observation that cells

generated during the sixth burst from VC #99 encounter greater delays than those from either the fourth or fifth bursts. In order to understand this somewhat counter-intuition we must first define a parameter we call the "EDF threshold". We define a cell's EDF threshold as the time at which no new arrivals at the switch will exit the switch before this cell exits the switch under EDF scheduling. The EDF threshold, T_i, for a given cell from VC_i with theoretical maximum delay bound, d_i^{max}, is given by

$$T_i = d_i^{max} - max\{d_j^{max}\} \tag{5.1}$$

where $d_j^{max} < d_i^{max}$.

Now the delay incurred by the sixth burst from VC #99 can be explained by the fact that between the time at which the last cell from VC #99's fifth burst exits the switch and the time at which the first cell of VC #99's sixth burst arrives at the switch, the delay values of the first burst of cells from all class B VCs have surpassed their EDF thresholds. Because of this, cells from the sixth burst of VC #99 must wait until all 128 of the class B first bursts have been served, giving them greater delays than the fifth burst of VC #99.

In this experiment, since each VC within a VC class has the same theoretical maximum delay bound and since we have 2 VC classes, we have 2 different theoretical maximum delay bounds, d_A^{max} and d_B^{max}, for class A and B VCs, respectively. Since $d_A^{max} < d_B^{max}$, we define the EDF threshold for all cells of class A, T_A, to be

$$T_A = 0.$$

Similarly, from Equation 5.1, we get

$$T_B = d_B^{max} - d_A^{max}.$$

It is also interesting to note another property of the EDF threshold. When a cell incurs less delay than its EDF threshold, then there were no cells inside the switch with a smaller theoretical maximum delay bound at the time of the cell's departure.

Because we observe an increase in a burst's delay from the previous burst's delay, we term this effect a "bounce". We consider the behavior of the sixth burst of cells from VC #99 in Figure 5.8 to be VC #99's second bounce, the first bounce occurring with the first burst of cells. Next, we designed controlled experiments to gain further insights into this second bounce phenomenon.

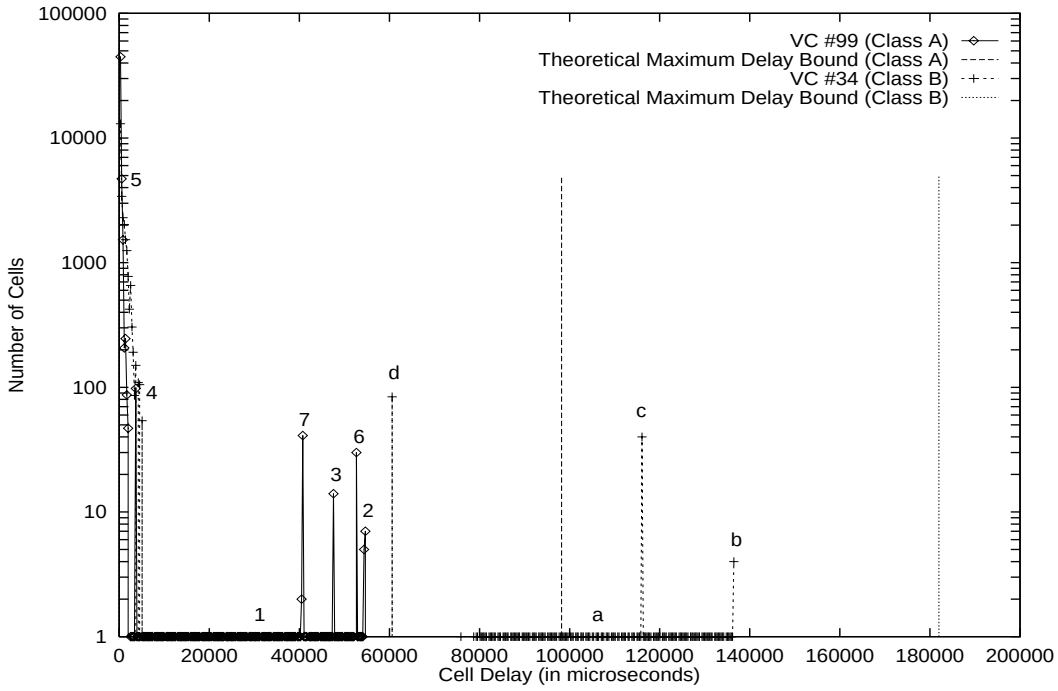

Figure 5.9: Manipulation of the second bounce in cell delays

In Figure 5.9 we show the results of a controlled experiment where VC #97's fifth pause period was manually set to 91,533 μsec while all other input parameters remained the same as in Figure 5.8. VC #99's fifth pause period was carefully chosen such that the first cell of VC #99's sixth burst arrived at the switch just after the delays of the first burst of all class B VCs had passed their EDF threshold. Clearly, this results in worst-case delays for the sixth burst. In Figure 5.9, we observe that the cell delays of VC #99's sixth burst increase significantly and are almost at the tail of the cell delay function.

Switch Output by Virtual Circuit Class

In order to more completely understand the behavior of heterogeneous VC classes during the warm-up period of simulations, we observed the cell output patterns for each class of VC over time at the beginning of our simulations. In each of these experiments we simulated 256 VCs using the three state model. The MBS for each VC was set at 8 KB, i.e. 171 cells, and the output link reservation level was set at approximately 97%. Two VC classes were used in these simulations with the same input parameters given for each VC within a class. As before, the ratio of class A SCRs to class B SCRs was 1.85:1.

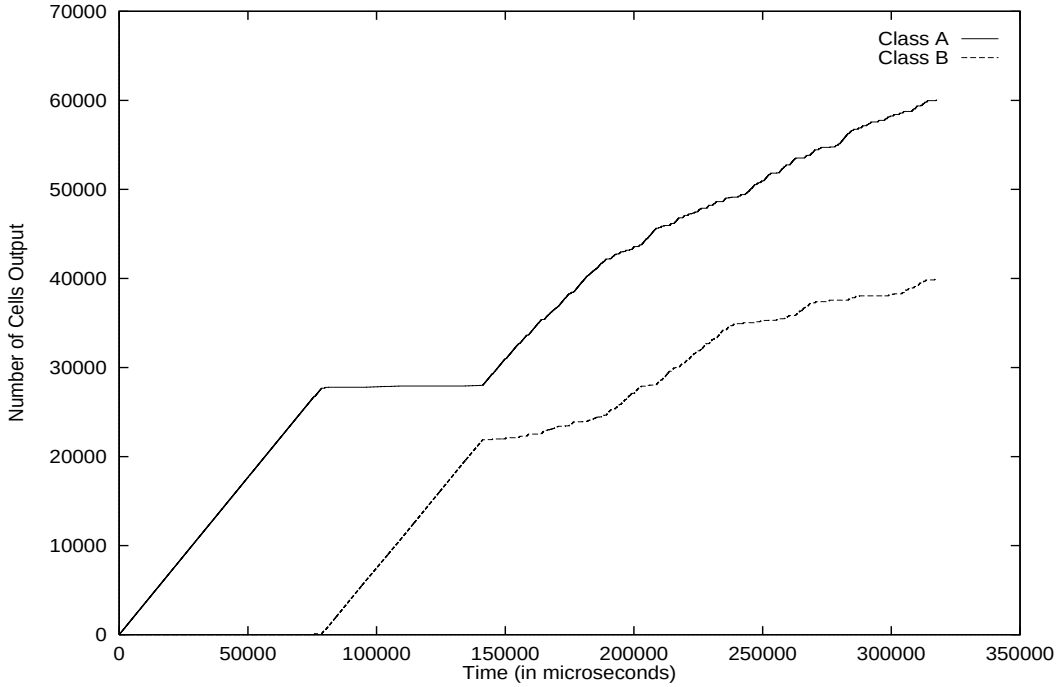

Figure 5.10: Cell output times by VC class with 64 input links

We simulated a mere 100,000 total cells since this proved to be sufficient for capturing the warm-up period. It was important to minimize the number of cells due to the enormous files output by these simulations.

The results of the two extreme experiments with 64 and 2 input links are shown in Figures 5.10 and 5.11, respectively. In both figures we observe pure class A traffic being output from the switch from time 0 until class B's EDF threshold, which occurs at time 83,777 μsec. This effect can be explained by a property of the EDF threshold which states that no class B cells may exit the switch before its EDF threshold as long as class A cells are present in the switch. For a more in depth discussion on the EDF threshold see p. 52.

After the EDF thresholds of the first class B cells have been reached, class B cells have greater priority than newly arriving class A cells; therefore, at this point we start to see an increase in the number of class B cells output. This class B increase occurs decidedly more sharply in the 64 input link case (see Figure 5.10) than in the 2 input link case (see Figure 5.11). This is attributed to the degree of multiplexing first discussed in Section 3.2.

In the 64 input link case, warm-up period cells incur larger delays inside the switch than do similar cells in the 2 input link case which incur larger delays inside a multiplexer.

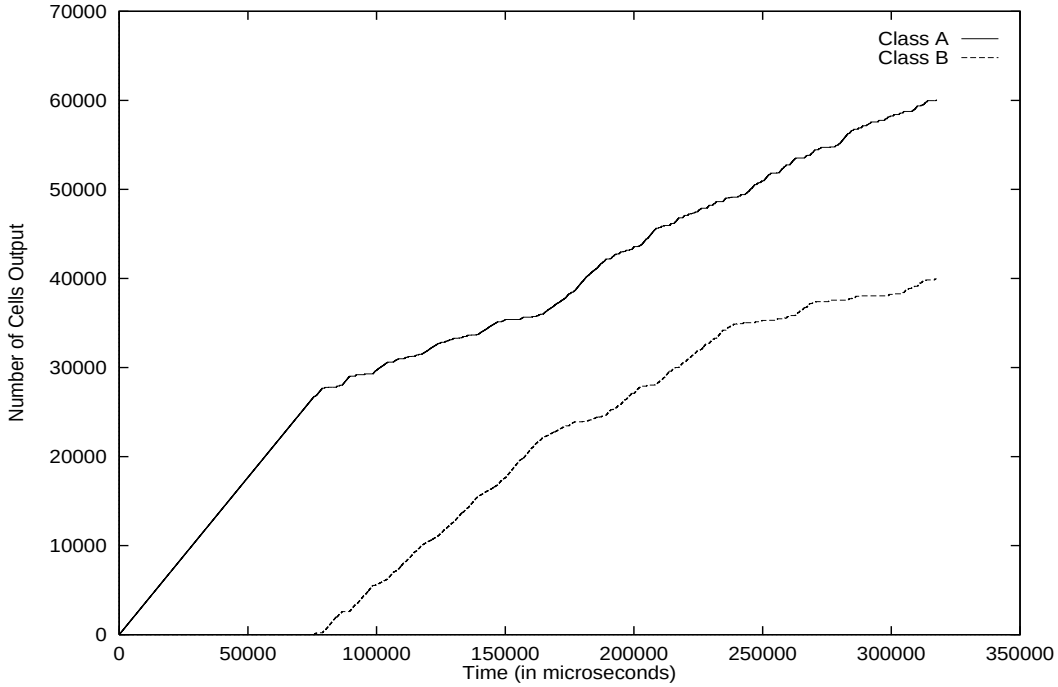

Figure 5.11: Cell output times by VC class with 2 input links

Therefore, at the time that the first class B cell departs from the switch there will be more class B cells inside the switch in the 64 input link case than in the 2 input link case. This, coupled with the observation that relatively few class A cells are in the switch at time 83,777 μsec (since up until that point in time only class A cells have been served), accounts for the steeper class B increase in Figure 5.10 than in Figure 5.11 beginning at time 83,777 μsec.

At approximately 140,000 μsec, we notice that the sharp increase in class A cells (more noticeably observed in Figure 5.10 for reasons given above) results from the build up of the class A cells which arrived while the initial class B cells were being transmitted. Of course, this is also attributed to having fewer class B cells in the switch since many class B cells were recently transmitted.

Finally, we observe that after approximately 200,000 μsec have elapsed, the switch has overcome the warm-up period and only minor variations between Figures 5.10 and 5.11 exist.

Note that in these figures, the rate of increase of one VC class is inversely proportional to the rate of increase of the other one, since only one cell is output from the switch

at any given point in time.

5.2 Persistent Sources

In addition to examining the behavior of cells using the three state model, we also observed the worst-case behavior of cells conforming to their leaky buckets. We explored two different varieties of this worst-case behavior. In the experiments reported in this section we abandoned the three state model and made every source "persistent", i.e. each source continually sent cells to the leaky bucket. The persistent source model is first described on p. 25.

With this persistent source behavior the leaky buckets accepted the entire first burst of cells and then admitted only one cell every $\frac{1}{\text{SCR}}$ interval of time thereafter. (An explanation how our simulated cell drops would actually be implemented in a leaky bucket is given in Section 4.1.2). The first MBS cells conform due to the fact that the BT is used in the limit of the SCR leaky bucket. Subsequent cells conform every $\frac{1}{\text{SCR}}$ thereafter, since this is the increment value of the SCR leaky bucket. Note that cells always conform to the PCR leaky bucket since the PCR is set to the time to transmit a cell at source link speed.

In both the persistent source experiments reported here, we simulated 256 VCs evenly distributed onto 64 input links with an MBS size of 8 KB, or 171 cells. As in previous experiments, we reserved 97% of the output link bandwidth in order to capture worst-case behavior.

5.2.1 The Aligned Case

In this section we consider the aligned case where all 256 sources begin transmission simultaneously. In the following section, we look at the case were these sources were randomly staggered.

We simulated 2 classes of VCs, 128 VCs per class, with an SCR ratio of class A VCs to class B VCs of 1.85:1.

The leaky bucket non-conformance rate of 170 out of each burst of 171 cells (after the initial burst) resulted in much longer simulation execution times for the same number of cells output in most other experiments (10,000,000 cells). Thus, we ran the experiment described here for 2,500,000 total cells output onto the output link. This number of cells was sufficiently large to fully satisfy our curiosity regarding the warm-up period and beyond.

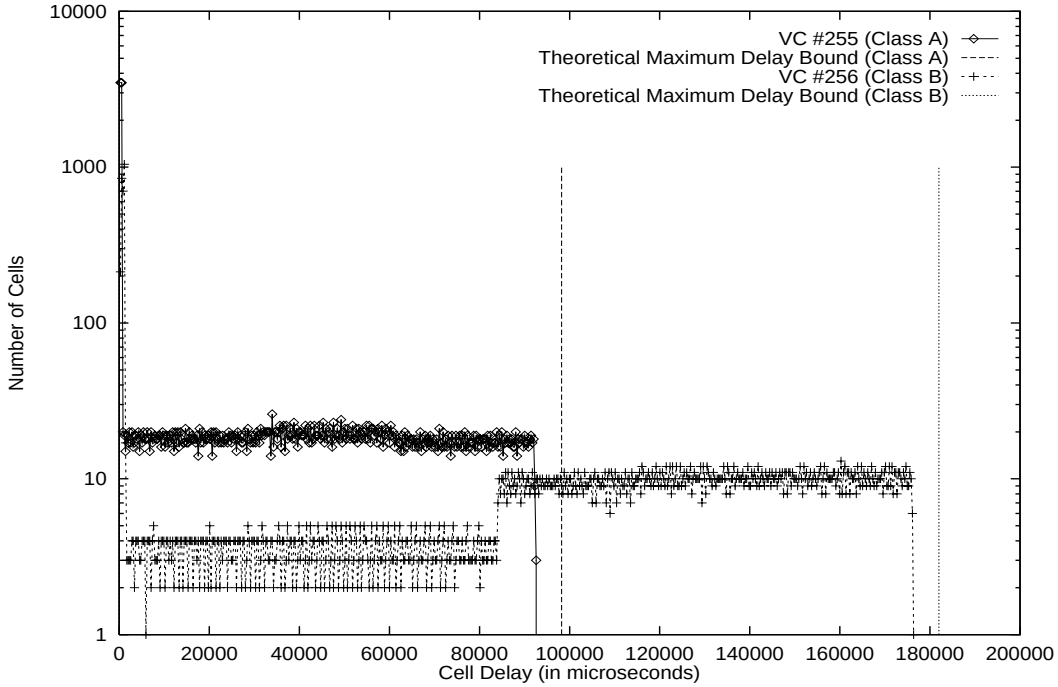

Figure 5.12: Cell delays with 256 persistent VCs in 2 VC classes on 64 input links in aligned mode

We show the cell delays from this experiment in Figure 5.12. In this figure we observe that cell delays come very close to reaching their theoretical maximum delay bounds. The reason these warm-up period cells incur much larger delays than in previous three state model experiments (see Figure 5.6 for a comparison), lies in the fact that all sources persistently send cells to the switch at SCR after their first burst. As compared to the three state model in which sources pause for a random amount of time between bursts, in our persistent source model the switch does not have as much time to output as many of the initial cells before new cells start arriving.

We note that the cells which incur more than approximately 1,000 μsec of delay contribute to the warm-up period of the switch. We note that after these cells have been output, the switch stabilizes and easily serves cells arriving at their SCR with very minimal delays. This is evidenced by the large peaks of cells in the extremely low cell delay bins.

We also observed that the worst-case delay of class B cells occur for the last cell of the first burst, since this cell must wait until many class A cells and all other class B cells in its same burst have been served. The worst-case delays for class A cells occur for those

cells which arrive at the switch just after the first class B cells reach their EDF threshold (see p. 52).

It is not coincidence that the worst-case VCs for each type plotted in Figure 5.12 are the largest VC numbers for each VC class. Deadline ties are broken arbitrarily in favor of lower VC numbers. Since the cell arrivals are deterministic within a class, VC #255 and VC #256 will lose the tie-breaker to other VCs in its VC class and incur slightly larger cell delays.

One final point to note in Figure 5.12 is that the deterministic nature of our persistent sources results in round-robin service within a VC class. Of course, this is not pure round-robin from a global perspective as there are varying degrees of overlap between cell delays in the two classes. This round-robin behavior results in the relatively even distributions of cells over large delay ranges. A distinct increase in the number of cells from VC #256 occurs precisely at VC #256's EDF threshold (83,777 μsec). Cells from VC #256 which incurred more delay than this threshold are those which were competing with the initial flood of class A cells. Those cells from VC #256 which incurred less delay than this threshold arrived after the initial flood of class A cells had exited the switch. They, therefore, competed largely with other class B cells as well as the relatively few class A cells which continued to arrive at SCR.

5.2.2 The Staggered Case

Next, we explore the behavior of the persistent source model when sources have staggered start times instead of aligned start times as in Figure 5.12. In this experiment sources were staggered such that at simulation initialization each VC waited for a random idle period (as described in the discussions of the three state model on p. 23). After this initial idle period, a source will persistently send cells to its leaky buckets as described in the previous experiment.

In this experiment we broadened our experiment to include four VC classes, lettered A through D. Each class contained 64 VCs and classes A, B, and C had SCR ratios with respect to class D given by 1.85:1, 1.60:1, and 1.24:1, respectively. This experiment was run for a total of 5,000,000 cells.

As can easily be seen in Figure 5.13, the maximum observed cell delays are so minimal that it is staggering! (pun intended) In this experiment, we see that cells do not

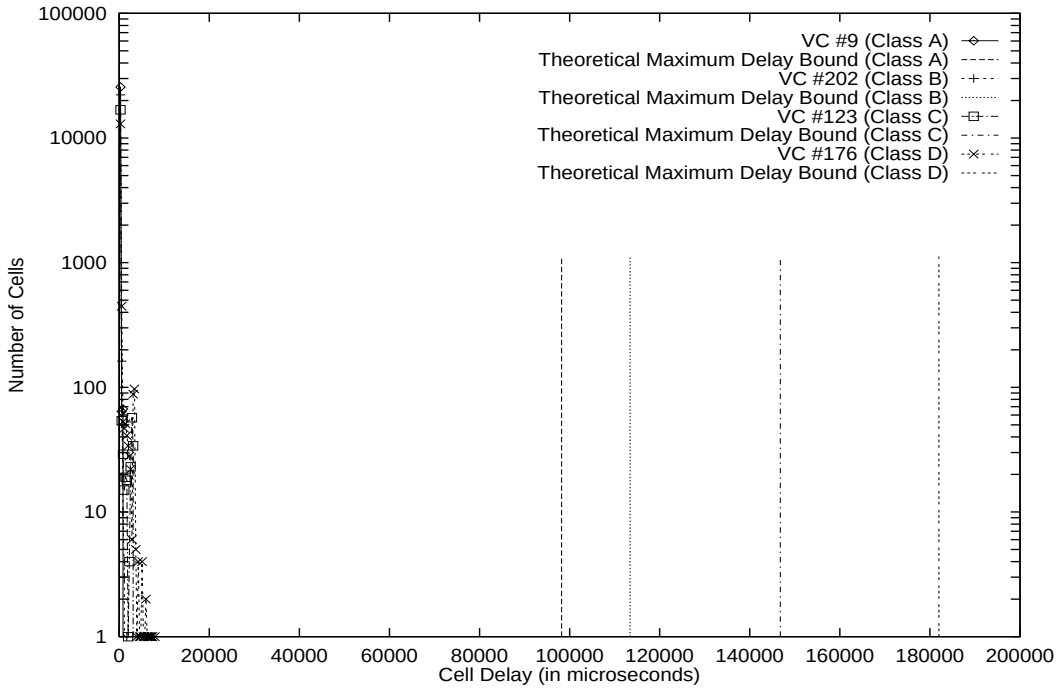

Figure 5.13: Cell delays with 256 persistent VCs in 4 VC classes on 64 input links in staggered mode

even come close to their theoretical maximum delay bounds. The reason for this is that the sources' initial pause periods are such that they seldom overlap. Therefore, each VC's first burst of cells must only compete with other VCs which had shorter initial pause periods and are likely transmitting cells at SCR. As we also observed in Figure 5.12, delays are very small when all VCs transmit cells at SCR.

This profound result explains more about the impact of worst-case behavior. Even though the leaky buckets are functioning with worst-case behavior, simply staggering the sources such that they do not have worst-case starting behavior results in drastic improvements in worst-case cell delays.

5.3 Repetitive Burst Sources

In this section we explore the second of the two worst-case leaky bucket behaviors mentioned previously in Section 5.2. We use the term "repetitive burst" source to describe sources with this behavior. The repetitive burst source model is first described on p. 26.

In the experiments in this section all sources infinitely repeat a pattern of trans-

mitting a burst of cells, then pausing for the time interval $\frac{\text{MBS}}{\text{SCR}} - \frac{\text{MBS}}{\text{PCR}}$. This pause period is the smallest time interval such that all cells will conform to their leaky buckets, since after this time has passed, the SCR leaky bucket will then be able to accept the next MBS burst of cells due to the BT used in the SCR leaky bucket's limit parameter (see Equation 4.4). Note that in this source model, all cells will be leaky bucket conformant.

In both the repetitive burst source experiments discussed here, we simulated 256 VCs evenly distributed onto 64 input links with an MBS size of 8 KB, or 171 cells. As in previous experiments, we reserved 97% of the output link bandwidth in order to capture worst-case results.

5.3.1 The Aligned Case

In this section we consider the aligned case where all 256 sources begin transmission simultaneously. In the following section, we look at the case were these sources were randomly staggered.

We simulated 2 classes of VCs, 128 VCs per class, where the SCR ratio of class A VCs to class B VCs was approximately 1.8:1. For consistency with Figure 5.12, 2,500,000 total cells were also output in this experiment.

Cell delays from this experiment are shown in Figure 5.14. We observe that as was the case in the persistent source model (see Section 5.2), the worst-case cell delays are relatively close to their theoretical maximum delay bounds.

In this experiment class A cells incur large delays when class B cells in the switch have deadlines such that the class B cells have higher priorities than the class A cells. The very sharp drop in VC #255 cells which incurred delays in excess of 60,000 μsec is evidence of this. At the time of most class A cell arrivals, few class B cells have surpassed their EDF threshold. Therefore, in these cases cells from VC #255 compete only with other class A cells, and are served in a round-robin fashion within the set of class A VCs. All class A cells send bursts simultaneously in this model, thus if no class B cells have a higher priority, the last cell in a burst from VC #255 will have a delay of approximately 59,800 μsec. This explains the drop in class A cells at approximately this delay value.

The behavior of the class B cells is not as clear-cut. It is rare that at the arrival time of a class B cell, few other cells are in the switch, since previous class B cells would likely be awaiting transmission. The few class B cells with very small delays support this

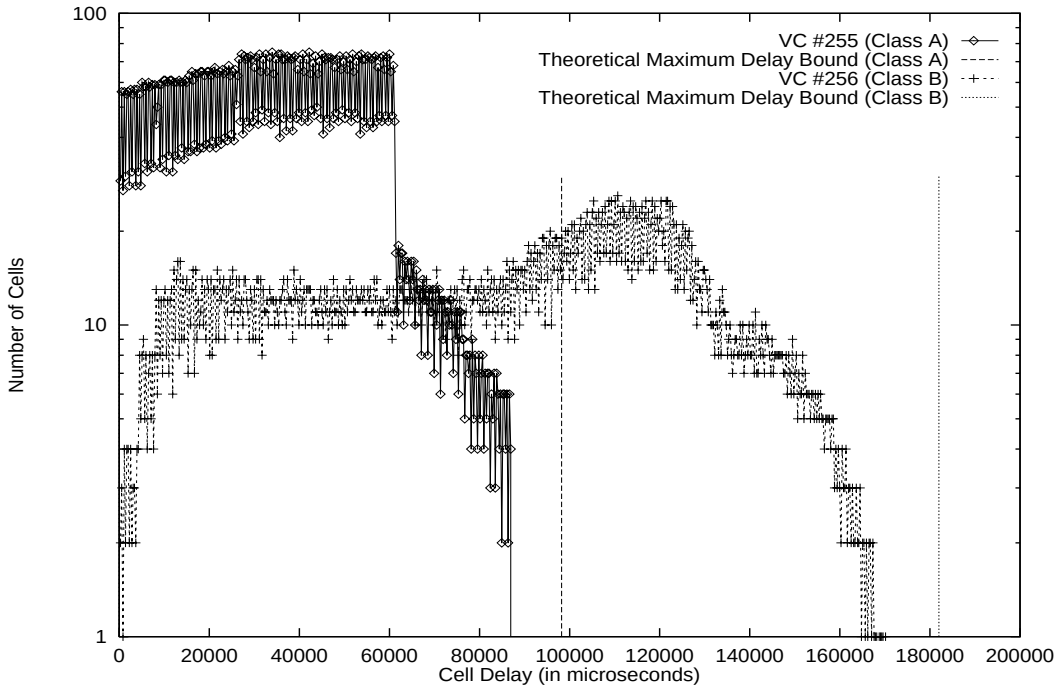

Figure 5.14: Cell delays with 256 repetitive burst VCs in 2 VC classes on 64 input links in aligned mode

intuition.

It is interesting to note that in the repetitive burst source model's aligned case, the switch is never able to overcome its warm-up period. Since cells arrive in bursts which have synchronized arrival times within each VC class, the switch is not able to service all of these cells before the next bursts arrive. Therefore, we note that unlike previous experiments, where post-warm-up period cells encountered small delays which resulted in very large peaks at the beginning of the delay functions, relatively few cells incur very minimal delays as there are no extremely large peaks at the beginning of the delay functions shown in Figure 5.14.

It is not coincidence that the worst-case VCs for each type plotted in Figure 5.14 are the largest VC numbers for each VC class. Deadline ties are broken arbitrarily in favor of lower VC numbers. Since the cell arrivals are deterministic within a class, VC #255 and VC #256 will lose the tie-breaker to other VCs in its VC class and incur slightly larger cell delays.

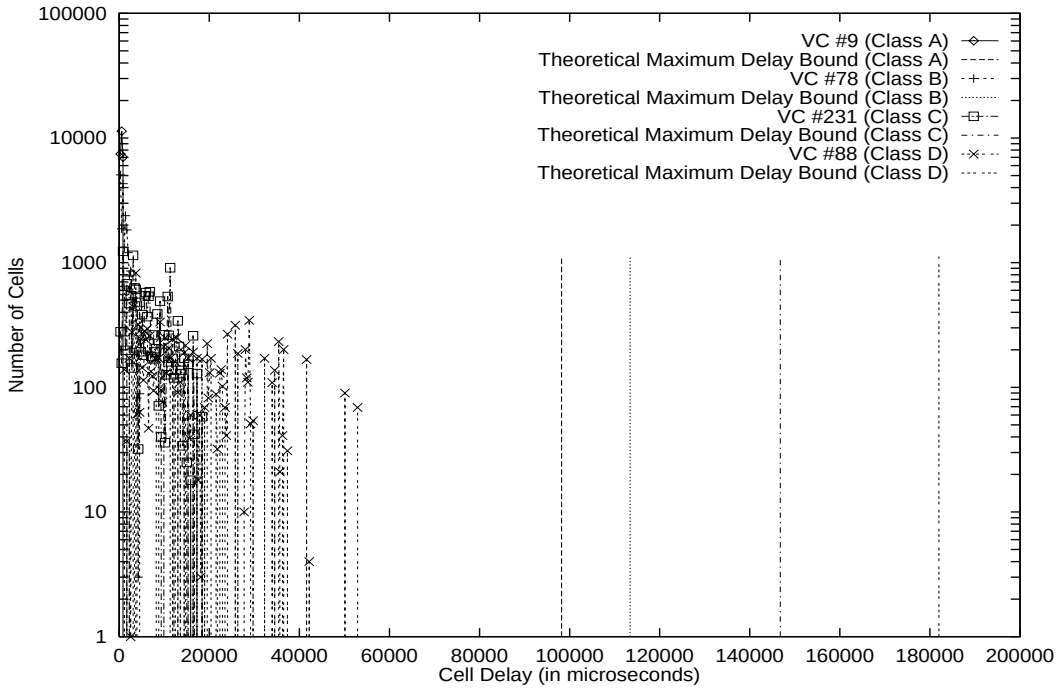

Figure 5.15: Cell delays with 256 repetitive burst VCs in 4 VC classes on 64 input links in staggered mode

5.3.2 The Staggered Case

Next, we experiment with staggering the sources' start times in the same fashion as our persistent source staggered experiment on p. 58. As in the persistent source staggered experiment, the same four VC classes and the same initial idle period random numbers were used in this experiment. The cell delays from this experiment can be seen in Figure 5.15.

In Figure 5.15 we make two important observations. First, we notice that the worst-case cell delays are not quite as favorable as in the persistent staggered experiment (see Figure 5.13). However, these delays are clearly much less than in the repetitive burst source aligned case (see Figure 5.14). As was seen in the persistent source experiments (see Section 5.2), the worst-case alignment of sources has a much more severe impact on the cell delays than does the worst-case leaky bucket behavior.

Secondly, we note that in Figure 5.15 we observe distinct cell peaks. Since all sources have the same MBS (171 cells) and the link speeds are all uniform at 155 Mbps, the sources will transmit bursts in a uniform amount of time. Also note that each source within a VC class will pause for the same amount of time $\left(\frac{\text{MBS}}{\text{SCR}} - \frac{\text{MBS}}{\text{PCR}}\right)$ between successive bursts.

Therefore, since sources start transmitting cells at random points in time, a source's bursts will be offset from the bursts from other sources in its VC class by the difference in their initial idle periods. Therefore, cells encounter less competition while inside the switch (thus have less delay) and a burst of cells from a given VC will tend to be transmitted nearly back-to-back on the output link. In general, when cells from the same source are output back-to-back, they encounter the same delay (since they arrived at the switch roughly back-to-back) and a peak on the cell delay graph results.

5.4 MPEG Traces

In this section we discuss our experimentation with MPEG video traces. A more in depth discussion of the specifics surrounding our MPEG trace data is outlined in Appendix D.

While we feel that experiments using the three state model described earlier in this chapter are reasonably good approximations of actual VBR traffic, there is no substitute for actual VBR video traces. We explored the results of actual MPEG video cell delays in order to benchmark our previous worst-case behavior.

We had at our disposal 18 of the MPEG traces from the FTP site listed in Appendix D. Unfortunately, in this experiment we needed more MPEG traces since we wanted to achieve a 97% reservation level and keep the output link speed at 155 Mbps as it was in all previous experiments. To solve this problem, we arbitrarily chose 6 MPEG traces to use twice each in our simulation. To prevent sources that used the same trace from having identical cell arrival times, each source began transmission at a random I-frame in the trace and looped to the beginning of the trace if the end of the trace was reached. Each trace file listed the size of the frames in bits. Since each trace contained 30 minutes of video, the trace files were sufficiently large for this experiment.

Next, we explain our calculations of the traffic parameters in this experiment. The PCR for each VC was no different than any of our other experiments—it was set to the time required to transmit a cell at our output link speed of 155 Mbps, 2.8312 μsec. The MBS for each VC was determined by finding the largest frame in the trace before running the simulation and then dividing this frame into cells, padding the last cell if necessary. The SCR was determined by multiplying the MBS by the frame rate. This SCR was such that all of the cells in a frame were leaky bucket conformant. The theoretical maximum delay

bound was calculated by dividing the MBS by the SCR and adding the time to transmit a cell at output link speed, according to Equation 4.3.

Note that this correlation between the MBS and SCR gives all cells the same theoretical maximum delay bound. Because of this, our EDF scheduler behaves as a FIFO would in this experiment.

In the MPEG trace experiment described in this section, we simulated 10,000,000 total cells from 25 different MPEG sources, each on its own input link into the switch. We chose the 25 MPEG sources such that their cumulative output link reservation level was approximately 97% as in previous experiments. A frame rate of 30 frames per second was used for all sources.

To capture worst-case source behavior in this experiment, all sources were aligned such that they all started transmitting their first I-frame at the same time. Also, there are two different assumptions that can be made concerning how the cells from a frame are transmitted. In one assumption called "smoothed mode" cells from a frame are evenly distributed through out each frame interval, i.e. the inverse of the frame rate. In the other assumption called "burst mode", cells from a frame are transmitted back-to-back soon as each frame is generated. In this experiment we chose burst mode because of our interest in worst-case behavior.

The cell delays of VCs with the worst-case (VC #2 and VC #23) and best-case (VC #19) behavior are shown in Figure 5.16. In this figure, we see that the majority of cells from these VCs experience very small delays and those cells with the largest delays are still relatively far from their theoretical maximum delay bounds. Clearly, the delays for the MPEG trace files are substantially less than any of our other worst-case source scenarios.

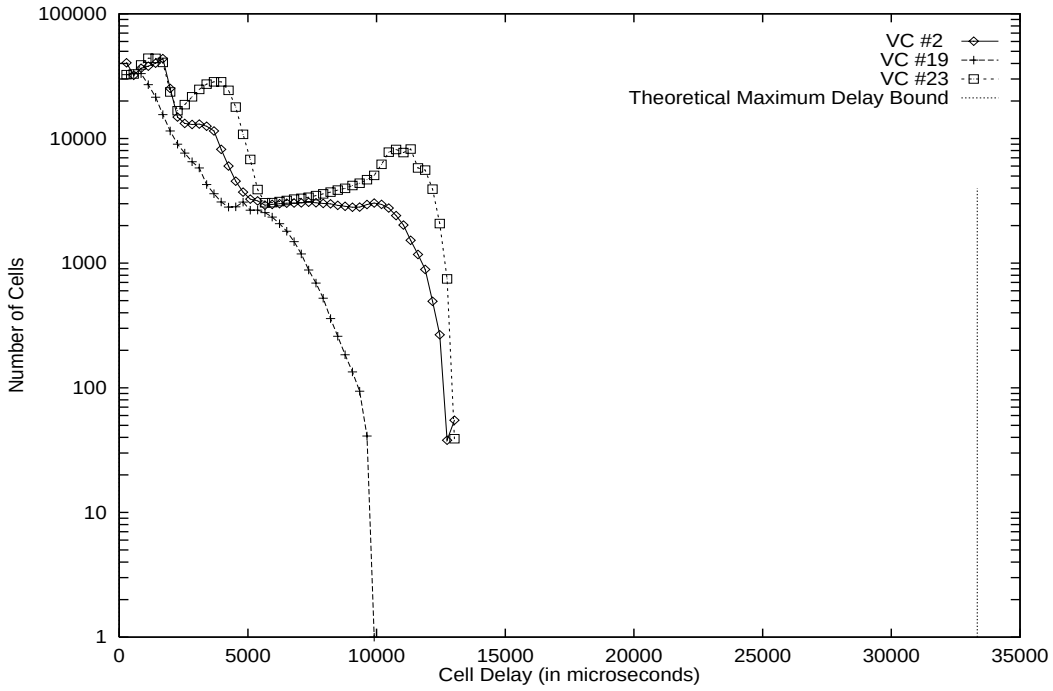

Figure 5.16: Cell delays with 25 MPEG video sources

Chapter 6

Summary and Future Work

6.1 Summary

The focus of our research has been the analysis of worst-case cell delays in an ATM switch using EDF scheduling. In particular, our interests are in the behavior of the tails of cell delay functions and their proximity to their theoretical maximum delay bounds. We have built a simulation program with which we closely studied cell delays and EDF scheduling behavior.

In our studies we simulated real-time VBR traffic using four different source models: the three state model, the persistent model, the repetitive burst model, and the MPEG trace model.

With the three state model, we showed that when sources are aligned to transmit simultaneously, worst-case cell delays occur while the switch is in its initial warm-up state. We also noted that after such time that these initial bursts of cells are served, cell delays are minimal, since large numbers of sources no longer transmit cells simultaneously.

We also examined the effects of the degree of multiplexing on cell delays inside the switch. We learned that during the switch's warm-up period, smaller cell delays result as the number of VCs multiplexed onto each link is increased. This is due to the fact that as the degree of multiplexing is increased, a larger proportion of the end-to-end delay shifts from occurring inside the switch to occurring inside the multiplexer.

We also researched providing delay guarantees as percentages of the theoretical maximum delay bounds for two different levels of output link reservation. We noted that worst-case cell delays decrease as the level of output link reservation decreases. We also

noted that large improvements in delays can result from minimal drops in delay percentages guaranteed. However, since these worst-case delays occur during the switch's warm-up period, these percentile results are skewed by the fact that we can provide arbitrarily favorable delay results by increasing the length of the simulation and thereby decreasing the effect of the warm-up period on the entire simulation.

We also experimented with multiple VC classes and noted that peaks may occur at the tails of the delay distribution functions as a result of competition inside the switch with cells both in the same and different VC class. We were able to manipulate the behavior of these peaks by manually setting some of the three state model parameters.

Also, in experimenting with multiple VC classes, we examined properties of a cell's EDF threshold—the time at which no newly arriving cells will depart the switch before this particular cell. The EDF threshold proved to be an important factor in the cell delays we observed.

In addition to the three state model, we also experimented with two source models with which we studied the limits of the leaky buckets. In the persistent source model, sources continuously send cells to the leaky buckets which accept the first burst of cells and then one cell at a rate equal to the SCR thereafter. In the repetitive burst model, sources send bursts of cells and then pause for just enough time such that the next burst of cells will be conformant to the leaky buckets.

With these two models, we explored both the aligned and staggered cases of cell transmission. With both source models, the staggered case resulted in drastic reductions in cell delay over the aligned case. In the aligned case, the worst-case delays were equally large for both of these source models. However, the repetitive burst model exhibited worse behavior in that very few of its cells have minimal delays, whereas in the persistent model, the post-warm-up period cells incurred very small delays. In the staggered case, the persistent model produced smaller worst-case cell delays than the repetitive burst model.

Finally, we reported our findings using actual MPEG video trace files. The worst-case cell delays observed using MPEG traces were far less than any of our other worst-case experiments.

6.2 Future Work

We feel that while we have made significant strides in analyzing cell behavior in an ATM switch using EDF scheduling, there still remains much more left to be considered.

While we have only considered a single-node network, we believe that another interesting area of research would be that of analyzing delays across multiple ATM switches implementing EDF scheduling. We believe that by performing traffic shaping at the output of each switch, favorable delay values may be obtained using EDF. Some challenges with this work may prove to be narrowing the scope of the parameter space. This would also include deciding onto which paths through the network flows should be placed.

Another future area of research would be an in depth analysis of the buffer requirements necessary for EDF scheduling in an ATM switch.

Finally, the development of CAC algorithms for an ATM switch using EDF would also be interesting future work.

Bibliography

[1] Jon C. R. Bennett and Hui Zhang. WF^2Q: worst-case fair weighted fair queueing. In *Proceedings of IEEE INFOCOM*, volume 1, pages 120–128, 1996.

[2] H. Jonathan Chao, Hsiling Cheng, Yau-Ren Jenq, and Daein Jeong. Design of a generalized priority queue manager for ATM switches. *IEEE Journal on Selected Areas in Communications*, 15(5):867–879, June 1997.

[3] The ATM Forum Technical Committee. Traffic management specification. Technical Report 95-0013R10, The ATM Forum, February 1996.

[4] R. L. Cruz. SCED+: efficient management of quality of service guarantees. To appear in Proceedings of IEEE INFOCOM '98, San Francisco, CA, March 1998.

[5] Khaled M. Fuad Elsayed and Harry G. Perros. A comparative performance analysis of call admission control schemes in ATM networks. To appear in the Journal of Computers and ISDN.

[6] Victor Firoiu, Jim Kurose, and Don Towsley. Efficient admission control for EDF schedulers. In *Proceedings of IEEE INFOCOM*, volume 1, pages 310–317, 1997.

[7] Leonidas Georgiadis, Roch Guérin, and Abhay Parekh. Optimal multiplexing on a single link: delay and buffer requirements. *IEEE Transactions on Information Theory*, 43(15):1518–1535, September 1997.

[8] Leonidas Georgiadis, Roch Guérin, Vinod Peris, and R. Rajan. Efficient support of delay and rate guarantees in an internet. *Computer Communication Review*, 26(4):106–116, October 1996.

[9] Leonidas Georgiadis, Roch Guérin, Vinod Peris, and Kumar N. Sivarajan. Efficient network QoS provisioning based on per node traffic shaping. *IEEE/ACM Transactions on Networking*, 4(4):482–501, August 1996.

[10] Pawan Goyal and Harrick M. Vin. Generalized guarantee rate scheduling algorithms: a framework. *IEEE/ACM Transactions on Networking*, 5(4):561–571, August 1997.

[11] Miki Hirano and Naoya Watanabe. Characteristics of a cell multiplexer for bursty ATM traffic. In *International Conference on Communications*, volume 1, pages 399–401, 1989.

[12] Takeshi Kawasaki, Miwako Nakashima, Toshio Soumiya, and Masafumi Katoh. A strategy of quality control on ATM switching network - quality control path (QCP). In *IEEE Global Telecommunications Conference*, volume 1, pages 432–436, 1996.

[13] Shih T. Liang and Maria C. Yuang. Departure process analysis for earliest-due-date scheduling discipline in ATM switches. *Computer Systems Science and Engineering*, 11(6):343–352, November 1996.

[14] Jörg Liebeherr, Dallas E. Wrege, and Domenico Ferrari. Exact admission control for networks with bounded delay service. *IEEE/ACM Transactions on Networking*, 4(6):885–901, December 1996.

[15] C.L. Liu and James W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the Association for Computing Machinery*, 20(1):46–61, January 1973.

[16] Abhay K. Parekh and Robert G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: the single-node case. *IEEE/ACM Transactions on Networking*, 1(3):344–357, June 1993.

[17] Abhay K. Parekh and Robert G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: the multiple node case. *IEEE/ACM Transactions on Networking*, 2(2):137–150, April 1994.

[18] Oliver Rose. Statistical properties of MPEG video traffic and their impact on traffic modeling in ATM systems. In *Proceedings of the 20th Annual Conference on Local Computer Networks*, volume 1, pages 397–406, 1995.

[19] Henning Schulzrinne, Jim Kurose, and Don Towsley. An evaluation of scheduling mechanisms for providing best-effort, real-time communication in wide-area networks. In *Proceedings of INFOCOM '94*, volume 3, pages 1352–1361, 1994.

[20] Marco Spuri and Giorgio C. Buttazzo. Efficient aperiodic service under earliest deadline scheduling. In *Proceedings of the Real-Time Systems Symposium*, pages 2–11, 1994.

[21] S. Wright. Delay bound simulation studies: VBR source behavior. Technical report, Fujitsu Network Communications, Inc., December 1997.

[22] Tarif Zein, Gerard Maral, and Dominique Seret. Performance modeling of a cell multiplexer for bursty ATM traffic. *International Journal of Electronics*, 71(6):967–975, December 1991.

[23] Hui Zhang and Edward W. Knightly. Providing end-to-end statistical performance guarantees with bounding interval dependent stochastic models. *Performance Evaluation Review*, 22(1):211–220, May 1994.

Appendix A

How to Run the Simulator

This appendix gives the reader a detailed account of how to run the simulator on UNIX. If the reader would like to use this simulator for similar research, the author may be contacted for a copy of the simulation program; however, the simulator will not otherwise be made publicly available.

The 12 C++ source files and 15 header files can be compiled and linked using the Makefile. To compile and link, simply type "make" in the directory where these files reside on a UNIX command line.

The simulator requires three arguments: "input file", "output file", and "plot file". These files can be arbitrarily named. The "input file" must exist before running the simulator. If the "output file" or "plot file" exist, then they will be overwritten, otherwise they will be created.

The "input file" is a text file which provides the simulator with information about the switch topology, VCs using the switch, and the VC's parameter set. A sample "input file" is shown below. Comments are any line beginning with a backward slash ("/"). Any number of blank lines may separate values. Parameters must be listed in the "input file" in the exact order shown below. A class of VCs is indicated by an integer starting with 1. For many parameters such as: source speed, packet size, idle period mean, etc., there must be as many values listed as there are VC classes.

```
// A sample input file for use with the simulator

// Total number of cells to transmit from the switch
   1000000
```

```
// Number of input ports
   4

// Topology of the VCs--the number of VCs per input link their classes
   1 2
   3 4
   1 2
   3 4

// Input buffers size
   5000

// Output buffer size
   5000

// Input and output link speed (in Mbps)
   149.76

// Speeds of the sources per VC class (in Mbps)
   149.76 149.76 149.76 149.76

// Packet sizes per VC class (in Bytes)
   8192 8192 8192 8192

// Mean idle period values per VC class (in microseconds)
   10000 23000 36000 50000

// Mean pause period values per VC class (in microseconds)
   1770 3150 4300 5500

// Mean number of packets in each active period per VC class
   100 83 66 50

// Source transmission start times per VC class
   0.0 0.0 0.0 0.0

// Traffic mode per VC class
   VBR 32 VBR VBR

// CDVT for PCR per VC class
   0.0 0.0 0.0 0.0

// CDVT for SCR per VC class
   1.0 1.0 1.0 1.0
```

```
// Verbose mode for variables
   True

// Verbose mode for sources
   False

// Verbose mode for leaky buckets
   False

// Verbose mode for the event list
   False

// Verbose mode for the ATM switch
   False

// Random number seed
   0
```

Note that in the above example, there are 8 VCs multiplexed onto 4 input links, 2 per link. Class 1 and class 2 VCs are multiplexed onto input link 1 and input link 3. Class 3 and class 4 VCs are multiplexed onto input link 2 and 4. Class 2 VCs are CBR traffic and the rest are VBR.

The "output file" is a text file which gives the user a detailed account of the results of the simulation. Various statistics surrounding the simulation parameters and results are recorded in this file. This same information is also printed on the screen as the simulator runs. A sample "output file" showing the results of the above sample "input file" is given below.

```
----------------------
 ATM SWITCH SIMULATOR
----------------------

MaxCells = 1000000
InputPortCount = 4
NumVCs = 8
VCsPerPort = {2, 2, 2, 2}
InputBufferLength = 5000
OutputBufferLength = 5000
InputPortSpeed = 149.76
InputPortCellXmitTime = 2.8312
Transfer time = 0.707799
```

```
SourceSpeed = {149.76, 149.76, 149.76, 149.76, 149.76, 149.76,
 149.76, 149.76}
SourceCellXmitTime = {2.8312, 2.8312, 2.8312, 2.8312, 2.8312
 2.8312, 2.8312, 2.8312}
PacketXmitTime = {484.135, 484.135, 484.135, 484.135, 484.135,
 484.135, 484.135, 484.135}
IdlePeriodMean = {10000, 23000, 36000, 50000, 10000, 23000,
 36000, 50000}
PausePeriodMean = {1770, 3150, 4300, 5500, 1770, 3150, 4300, 5500}
NumPacketsMean = {100, 83, 66, 50, 100, 83, 66, 50}
StartXmitAtTime = {0, 0, 0, 0, 0, 0, 0, 0}
PeakCellInterval = {2.8312, 32, 2.8312, 2.8312, 2.8312, 32, 2.8312, 2.8312}
SustainedCellInterval = {13.6634, 32, 30.7862, 40.1996, 13.6634, 32,
 30.7862, 40.1996}
MaxBurstSize = {171, 1, 171, 171, 171, 1, 171, 171}
CellDelayVarTolerancePCR = {0, 0, 0, 0, 0, 0, 0, 0}
CellDelayVarToleranceSCR = {1, 1, 1, 1, 1, 1, 1, 1}
DelayBound = {2339.27, 34.8312, 5267.27, 6876.97, 2339.27, 34.8312,
 5267.27, 6876.97}
TCPPacketSize = {8192, 8192, 8192, 8192, 8192, 8192, 8192, 8192}
CellsPerPacket = {171, 171, 171, 171, 171, 171, 171, 171}
Seed = 0
verbose_Variables = true
verbose_Source = false
verbose_LB = false
verbose_EventList = false
verbose_Switch = false
-------------------------------

---------------------------
           CAC
---------------------------
VC #1 reserves 31.0319 Mbps
VC #2 reserves 13.25 Mbps
VC #3 reserves 13.7724 Mbps
VC #4 reserves 10.5474 Mbps
VC #5 reserves 31.0319 Mbps
VC #6 reserves 13.25 Mbps
VC #7 reserves 13.7724 Mbps
VC #8 reserves 10.5474 Mbps
---------------------------
Total reserved = 137.203 Mbps

Input Link Speed = 149.76 Mbps
```

```
Percentage reserved = 91.6155 %

Cell count of 1000000 has been reached.
0 cells were dropped inside of the switch.
Average delay of cells is 163.413
Output link utilization is 67.5743 %
Delay Bin Width = 100 times Cell Transmission Time.
InputBuffer maximum sizes = {1, 1, 1, 1, 1, 1, 1, 1}
OutputBuffer maximum size = 747
VC maximum delay = {646.221, 9.15625, 2613.78, 4049.32, 649.052, 9.20139,
 2202.07, 4052.15}
Greatest observed VC maximum delay for VC Type 1 = 649.052
Greatest observed VC maximum delay for VC Type 1 occurs on VC #s = {5}
Greatest observed VC maximum delay for VC Type 2 = 9.20139
Greatest observed VC maximum delay for VC Type 2 occurs on VC #s = {6}
Greatest observed VC maximum delay for VC Type 3 = 2613.78
Greatest observed VC maximum delay for VC Type 3 occurs on VC #s = {3}
Greatest observed VC maximum delay for VC Type 4 = 4052.15
Greatest observed VC maximum delay for VC Type 4 occurs on VC #s = {8}
Number of Cells Per VC  = {213758, 130930, 83679, 65779, 215637, 130930,
 91103, 68184}
```

The "plot file" is a text file which gives the user cell delays on a per-VC basis. This generated file can be used to produce the cell delay functions shown in Chapter 5. The first column of data in this file represents the right (greatest) edge of the delay bin given in μsec. Data in subsequent columns represent the number of cells which were captured in each bin on a per-VC basis starting with VC #1. A sample "plot file" showing the results of the above sample "input file" is shown below.

```
283.12 202986 131836 54606 29288 199032 131836 50770 32630
566.239 13333 0 14711 11626 12688 0 13219 8717
849.359 395 0 12332 7579 294 0 10260 6456
1132.48 0 0 5607 6360 0 0 6287 4898
1415.6 0 0 3087 3010 0 0 3422 4030
1698.72 0 0 932 2318 0 0 1187 3316
1981.84 0 0 258 2112 0 0 485 1554
2264.96 0 0 63 1044 0 0 166 1318
2548.08 0 0 98 931 0 0 62 736
2831.2 0 0 0 672 0 0 0 472
3114.32 0 0 0 146 0 0 0 258
3397.44 0 0 0 112 0 0 0 81
3680.56 0 0 0 116 0 0 0 268
```

Note in the above "plot file" example, the bin width is 283.12 μsec. Also note that the vast majority of cells incur minimal delays.

Finally, the simulator can be run by typing:

```
unix% simulator inputfilename outputfilename plotfilename
```

Appendix B

Simulator Utilities

This appendix describes various small utility programs which complement the simulation program described in Appendix A.

B.1 CAC

The CAC program allows a user to view the CAC statistics for a given parameter set without actually running the simulation. The CAC program itemizes the amount of bandwidth needed by each VC in the simulation and computes the sum of these reservation amounts as a percentage of the input and output link speed. This same CAC program is run by the simulator at the beginning of each simulation.

The necessary .cpp and .h files can be compiled and linked on UNIX using the makefile "MakefileCAC". To compile and link this utility, type the following in the directory where these files reside on a UNIX command line:

```
unix% make -f MakefileCAC
```

The CAC program displays the results to the terminal. This utility takes one or two parameters. The first parameter is required and contains the name of an "input file", specified with the format described in Appendix A. An optional "output file" may also be specified as the second parameter. If an "output file" is specified, the same data printed to the screen is also printed to this file. If this file exists prior to execution, it is overwritten; otherwise, it is created. To run the CAC utility type:

```
unix% CAC inputfilename
```

or

```
unix% CAC inputfilename outputfilename
```

B.2 ExtractColumns

This utility program is used for condensing the simulation-generated "plot file" as described in Appendix A. The need for this utility arose out of the limitations of gnuplot, our graphing program on UNIX. We ran into obstacles when we tried plotting "plot files" with a large number of columns, like 257. The ExtractColumns utility condenses the simulation-generated "plot file" by saving only the columns requested by the user into a separate file—a "condensed plot file".

ExtractColumns currently supports the extraction of at least one, but not more than four columns of data. However, this feature could easily be expanded to support more columns. The first column of data in a "plot file" always represents the right edge of a cell bin (see Appendix A) and is always copied to the resulting "condensed plot file".

ExtractColumns may have anywhere between three and six arguments depending on how many columns are desired. The first parameter is required and is an integer representing a VC number. This number corresponds to a column in the "plot file" beginning with zero. Three additional VC numbers may optionally be specified. The next to last parameter given is the name of a valid "plot file". The "plot file" must exist prior to executing this utility. The last parameter is the name of a "condensed plot file". If the "condensed plot file" exists, it will be overwritten; otherwise, it is created.

An example use of the ExtractColumns utility is:

```
unix% ExtractColumns 73 42 203 plotfilename condensedplotfilename
```

This example extracts the 1^{st}, 74^{th}, 43^{rd}, and 204^{th} columns from the "plot file". Note, the order of the integers given in arguments one, two, and three is irrelevant.

The ExtractColumns utility may also be desired by isolating VCs of interest if disk space is at a premium.

B.3 GetPercentile

This utility is used in determining in which cell bin a given percentile of cells cumulatively exist. This was used in studying percentile guarantees in our simulations.

GetPercentile requires a percentile ranging from 0 to 100 and a valid "plot file" name (see Appendix A) as parameters. The utility passes through the "plot file" once counting the total number of cells on a per-VC basis. Next, the utility makes a second pass through the "plot file", this time counting cells until the input percentile is reached. Once this percentile is reached, the right edge of each cell bin (see Appendix A) is stored for each VC. These bin edges are reported for each VC. Finally, the maximum and minimum bin edges over all VCs at this particular percentile are reported.

The following example shows how to report the 99.9^{th} percentile using this utility:

```
unix% GetPercentile 99.9 plotfilename
```

Appendix C

Simulation Parameters

This appendix provides additional information regarding the simulations discussed in Chapter 5. In Table C.1, we summarize the input parameters which were varied in our simulations. In Table C.2, we provide the output link utilization percentages for relevant simulations.

Table C.1: Selected simulation input parameters

Figure	Source Model	Start Mode	Input Links	VCs	Cells	VC Classes	Reserved
5.1	Three State	Aligned	64	256	10,000,000	1	97%
5.2	Three State	Aligned	2	256	10,000,000	1	97%
5.3	Three State	Aligned	64	256	10,000,000	1	97%
5.4	Three State	Aligned	1...64	256	10,000,000	1	97%
5.5	Three State	Aligned	1...64	256	10,000,000	1	65%
5.6	Three State	Aligned	32	256	10,000,000	2	97%
5.7	Three State	Aligned	32	256	10,000,000	2	97%
5.8	Three State	Aligned	16	256	10,000,000	2	97%
5.9	Three State	Aligned	16	256	10,000,000	2	97%
5.10	Three State	Aligned	64	256	100,000	2	97%
5.11	Three State	Aligned	2	256	100,000	2	97%
5.12	Persistent	Aligned	64	256	2,500,000	2	97%
5.13	Persistent	Staggered	64	256	5,000,000	4	97%
5.14	Repetitive Burst	Aligned	64	256	2,500,000	2	97%
5.15	Repetitive Burst	Staggered	64	256	5,000,000	4	97%
5.16	MPEG	Aligned	25	25	10,000,000	1	97%

Table C.2: Output link utilization percentages

Figure	Utilization
5.1	61.64%
5.2	61.64%
5.6	61.45%
5.7	61.45%
5.8	61.45%
5.9	61.45%
5.12	99.99%
5.13	96.00%
5.14	99.99%
5.15	95.59%
5.16	10.46%

Appendix D

MPEG Traces

The MPEG traces were created by Oliver Rose [18]. The MPEG encoder parameters used in his collection of traces are listed in Table D.1.

The MPEG traces and additional information can be found at the following FTP address: ftp-info3.informatik.uni-wuerzburg.de/pub/MPEG/

A statistical analysis of these traces was performed by the Real-Time Communication team at North Carolina State University. The statistics are given at the following URL: www2.ncsu.edu/eos/service/ece/project/rtcomm/ewfulp/WWW/

Table D.1: MPEG encoder parameters used in generating traces

Parameter	Value
Encoder Input	384 x 288 pel
Color Format	YUV (4:1:1, resolution of 8 bits)
Quantization Values	I=10, P=14, B=18
Pattern	IBBPBBPBBPBB
GOP Size	12
Motion Vector Search	'Logarithmic' / 'Simple'
Reference Frame	'Original'
Slices	1
Vector/Range	half pel / 10